Structural Analysis of Multimode DAE Systems: summary of results

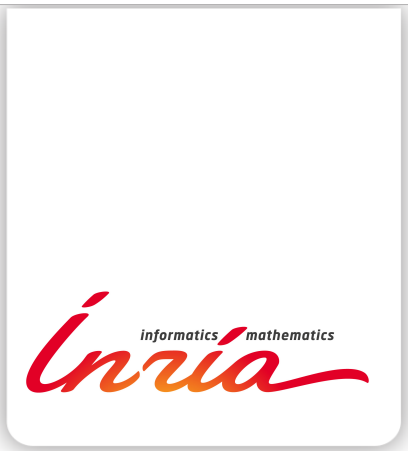
Albert Benveniste, Benoît Caillaud, Mathias Malandain

Structural Analysis of Multimode DAE Systems: summary of results

Albert Benveniste, Benoit Caillaud, Mathias Malandain

RESEARCH REPORT
N° 9387
January 2021
Project-Team Hycomes

ISSN 0249-6399 ISRN INRIA/RR--9387--FR+ENG



Structural Analysis of Multimode DAE Systems: summary of results

Albert Benveniste*, Benoit Caillaud, Mathias Malandain

Project-Team Hycomes

Research Report n° 9387 — January 2021 — 27 pages

This work was supported by the FUI ModeliScale DOS0066450/00 French national grant (2018-2021) and the Inria IPL ModeliScale large scale initiative (2017-2021, https://team.inria.fr/modeliscale/).

* All authors are with Inria-Rennes, Campus de Beaulieu, 35042 Rennes cedex, France; email: surname.name@inria.fr

RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

Abstract: Modern modeling languages for general physical systems, such as Modelica, Amesim, or Simscape, rely on Differential Algebraic Equations (DAEs), i.e., constraints of the form $f(x', x, u) = 0$. This drastically facilitates modeling from first principles of the physics, as well as model reuse. In recent works [2, 3], we presented the mathematical theory needed to establish the development of compilers and tools for DAE-based physical modeling languages on solid mathematical grounds.

At the core of this analysis sits the so-called *structural analysis*, whose purpose, at compile time, is to either identify under- and overspecified subsystems (if any), or to rewrite the model in a form amenable of existing DAE solvers, including the handling of mode change events. The notion of "structure" collects, for each mode and mode change event, the variables and equations involved, as well as the *latent equations* (additional equations redundant with the system), needed to prepare the code submitted to the solver. The notion of DAE *index* (the minimal number of times any equation has to be possibly differentiated) is part of this structural analysis.

This report complements [2, 3] by collecting all the needed background on structural analysis. The body of knowledge on structural analysis is large and scattered, which also motivated us to collect it in a single report.

We first explain the primary meaning of structural analysis of systems of equations, namely the study of their regularity or singularity in some generic sense. We then briefly review the body of graph theory used in this context. We develop some extensions, for which we are not aware of any reference, namely the structural analysis of systems of equations with existential quantifiers.

For the structural analysis of DAE systems, we focus on John Pryce's Σ-method, that we both summarize and extend to non-square systems.

The uses of these tools and methods in [2, 3] are highlighted in this report.

Key-words: structural analysis, differential-algebraic equations (DAE), multi-mode systems, variable-structure models

Analyse Structurelle des Systèmes de DAE multimodes: recueil des résultats de base

Résumé : Les langages modernes de modélisation de systèmes physiques, tels que Modelica, Amesim ou Simscape, s'appuient sur des Équations Algébro-Différentielles (DAE, de l'anglais *Differential-Algebraic Equations*), c'est-à-dire des contraintes de la forme $f(x', x, u) = 0$. La modélisation à partir des premiers principes de la physique, ainsi que la réutilisation de modèles, sont facilitées par cette approche. Dans des travaux récents [2, 3], nous présentons la théorie mathématique requise pour le développement de compilateurs et d'outils pour les langages de modélisation à base de DAE sur des bases mathématiques solides.

Cette analyse s'appuie sur l'*analyse structurelle*, dont le but, à la compilation d'un modèle, est soit d'identifier d'éventuels sous-systèmes sous- et sur-déterminés, soit de réécrire le modèle en vue de son traitement par des solveurs de DAE existants, en prenant en compte les événements de changements de mode. La notion de "structure" rassemble, pour chaque mode et chaque changement de mode, les équations et les variables impliquées, ainsi que les *équations latentes* (des équations supplémentaires redondantes), requises pour construire le code soumis au solveur numérique. La notion d'*index* d'un système de DAE, lié aux nombres de différentiations successives devant être appliquées à ses équations, est partie intégrante de cette analyse structurelle.

Le présent rapport vient en complément de [2, 3], en rassemblant toutes les connaissances requises en analyse structurelle. Ce travail a également été motivé par le fait que ce vaste corpus de connaissances est éparpillé dans la littérature.

Nous expliquons d'abord le sens premier de l'analyse structurelle de systèmes d'équations, à savoir, l'étude de leur régularité ou singularité dans un sens générique. Nous passons ensuite en revue les éléments de théorie des graphes utilisés dans ce contexte. Nous développons également une extension qui est, à notre connaissance, inédite dans la littérature sur le sujet : l'analyse structurelle de systèmes d'équations contenant des quantificateurs existentiels.

Pour l'analyse structurelle de systèmes de DAE, nous nous focalisons sur la Σ-méthode de J. Pryce, qui est résumée puis étendue à des systèmes non-carrés.

Les utilisations de ces outils et méthodes dans [2, 3] sont mises en évidence dans ce rapport.

Mots-clés : analyse structurelle, équations algébro-différentielles (DAE), systèmes multi-mode, modèles à structure variable

Contents

1 Introduction

Modern modeling languages for general physical systems, such as Modelica, Amesim, or Simscape, rely on Differential Algebraic Equations (DAEs), i.e., constraints of the form $f(x', x, u) = 0$. This drastically facilitates modeling from first principles of the physics, as well as model reuse. In recent works [2, 3], we presented the mathematical theory needed to establish the development of compilers and tools for DAE-based physical modeling languages on solid mathematical grounds.

At the core of this analysis sits the so-called *structural analysis*, whose purpose, at compile time, is to either identify under- and overspecified subsystems (if any), or to rewrite the model in a form amenable of existing DAE solvers, including the handling of mode change events. The notion of "structure" collects, for each mode and mode change event, the variables and equations involved, as well as the *latent equations* (additional equations redundant with the system), needed to prepare the code submitted to the solver. The notion of DAE *index* (the minimal number of times any equation has to be possibly differentiated) is part of this structural analysis. The body of knowledge on structural analysis is large and scattered, which motivated us to collect it in a single report. This report complements [2, 3] by collecting all the needed background on structural analysis. The uses of these tools and methods in [2, 3] are highlighted in this report.

Structural analysis of systems of equations: We first explain the primary meaning of the structural analysis of systems of equations, namely the study of their regularity or singularity in some generic sense. Structural information about a system of equations can be summed up by its incidence matrix; in particular, a square system yields a square matrix. If, after some permutation of row and columns, this matrix only has non-zero diagonal entries, then we say that this matrix, as well as the underlying system of equations, are structurally regular: this notion is repeatedly used in the Modelica community.

However, the most commonly used structural representation of a system of equations is actually the bipartite graph characterizing the pattern of the incidence matrix (i.e., the (i, j)-locations of its non-zero entries). Structural regularity or singularity, as well as derived notions, are then checked on this bipartite graph. As a matter of fact, the systems of equations under study are often sparse (i.e., their incidence matrices have a small proportion of non-zero entries), and the association of graphs with sparse matrices is pervasive in linear algebra for high performance computing [12, 13]. We briefly review the body of graph theory used in this context. We recall in particular the Dulmage-Mendelsohn decomposition of a bipartite graph, which is the basis for structuring a general (non-square) matrix into its overdetermined, regular, and underdetermined blocks, structurally. This body of knowledge in linear algebra is referred to as *structural analysis* of matrices. Structural analysis extends to systems of numerical equations, by considering the Jacobian of the considered system.

The link between numerical regularity and structural regularity belongs to the folklore of linear algebra; however, its understanding is critical for a correct use of structural analysis for systems of numerical equations. We were, however, unable to find a reference where related results were detailed; we propose such results here.

We also develop some useful extensions, for which we are not aware of any reference, particularly the structural analysis of systems of equations with existential quantifiers.

Structural analysis of DAE systems: We focus on DAE and summarize John Pryce's Σ-method for structural analysis [17], and we present an extension of this method to non-square DAE systems. We also include the structural form of Campbell and Gear's method of *differential arrays* [7], which we complement with its discrete-time counterpart, namely "difference arrays". These methods rely on the structural analysis of systems of equations with existential quantifiers.

2 Use of structural analysis on a toy example

This section is a summary of Sections 1.2.1 and 2 of [3], to which the reader is referred for further details.

2.1 An ideal clutch

Our example is an idealized clutch involving two rotating shafts with no motor or brake connected (Figure 1). We assume this system to be closed, with no interaction other than explicitly specified.

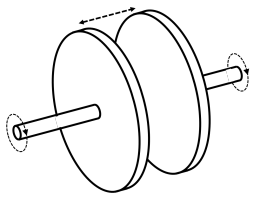


Figure 1: An ideal clutch with two shafts.

The dynamics of each shaft $i = 1, 2$, is modeled by $\omega_i' = f_i(\omega_i, \tau_i)$ for some, yet unspecified, function f_i, where ω_i is the angular velocity, τ_i is the torque applied to shaft i, and ω_i' denotes the time derivative of ω_i. Depending on the value of the input Boolean variable γ, the clutch is either engaged ($\gamma = \text{T}$, the constant "true") or released ($\gamma = \text{F}$, the constant "false"). When the clutch is released, the two shafts rotate freely: no torque is applied to them ($\tau_i = 0$). When the clutch is engaged, it ensures a perfect join between the two shafts, forcing them to have the same angular velocity ($\omega_1 - \omega_2 = 0$) and opposite torques ($\tau_1 + \tau_2 = 0$). Here is the model:

$$\left\{\begin{array}{lll} & \omega_1' = f_1(\omega_1, \tau_1) & (e_1) \\ & \omega_2' = f_2(\omega_2, \tau_2) & (e_2) \\ \texttt{if } \gamma \texttt{ then} & \omega_1 - \omega_2 = 0 & (e_3) \\ \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \\ \texttt{if not } \gamma \texttt{ then} & \tau_1 = 0 & (e_5) \\ \texttt{and} & \tau_2 = 0 & (e_6) \end{array}\right. \tag{1}$$

When $\gamma = \text{T}$, equations (e_3, e_4) are active and equations (e_5, e_6) are disabled, and vice-versa when $\gamma = \text{F}$. If the clutch is initially released, then, at the instant of contact, the relative speed of the two rotating shafts jumps to zero; as a consequence, an impulse is expected on the torques.

2.2 Analyzing the two modes separately

This model yields an ODE system when the clutch is released ($\gamma = \text{F}$), and a DAE system of index 1 when the clutch is engaged ($\gamma = \text{T}$), due to equation (e_3) being active in this mode. The basic way of generating simulation code for the engaged mode ($\gamma = \text{T}$) is to add, to the model in this mode, the *latent equation* (e_3'):

$$\left\{\begin{array}{ll} \omega_1' = f_1(\omega_1, \tau_1) & (e_1) \\ \omega_2' = f_2(\omega_2, \tau_2) & (e_2) \\ \omega_1 - \omega_2 = 0 & (e_3) \\ \omega_1' - \omega_2' = 0 & (e_3') \\ \tau_1 + \tau_2 = 0 & (e_4) \end{array}\right. \tag{2}$$

By doing so, we can apply the following basic policy:

> Regard System (2) as a system of equations in which the dependent variables are the leading variables of System (2), i.e., the torques τ_i and the accelerations ω_i', and consider the ω_i''s as *dummy derivatives,* i.e., fresh variables not related to the velocities ω_i; ignore equation (e_3) and solve the four remaining equations for the dependent variables, using an algebraic equation solver. (3)

This policy amounts to, first, bringing the DAE system back to an "ODE-like" form, then solving it as such. In this case, once the latent equation (e_3') was added, the Jacobian matrix of the system

(e_1, e_2, e'_3, e_4) with respect to its leading variables was invertible. This is the classical approach to DAE simulation via "index reduction"—having differentiated (e_3) once indicates that System (2) has index 1.

Basic tool 1 *The current practice in DAE-based modeling languages is that adding latent equations is performed on the basis of a* structural analysis, *which is a semi-decision procedure using graph theoretic criteria, of much lower computational cost than that of studying the numerical invertibility of the Jacobian. We provide the bases for structural analysis of DAEs in Sections 3 and 4 of this report.*

The analysis of the system above in both its modes is classical. The handling of mode changes, however, is not, due to the occurrence of impulsive torques in the "released → engaged" transition. Also, the nature of time differs between the study of modes and that of mode changes. Each mode, "released" or "engaged", evolves in *continuous-time* according to a DAE system. In contrast, the mode changes are events at which restart values for states must be computed from states before the change, in one or several *discrete-time* transitions—note that such restart transitions are not explicitly specified in System (1).

2.3 Analyzing mode changes

Let us focus on the "released → engaged" transition, where torques are impulsive.

To unify the views of time, we map the different kinds of time to a same discrete-time line, by discretizing the real time line $\mathbb{R}_{\geq 0}$ using an *infinitesimal* time step ∂. By doing so, the DAEs acting within the two "released" or "engaged" modes, get approximated with an *infinitesimal* error. By "infinitesimal", we mean: "smaller in absolute value than any non-zero real number". All of this can be mathematically formalized by relying on *Nonstandard Analysis* [18, 9], see [2, 3] for details. In particular, System (1) is mapped to discrete-time by using the discrete-time line

$$\mathbb{T} \quad =_{\text{def}} \quad \{n\partial \mid n = 0, 1, 2, \dots\} \tag{4}$$

and replacing in it all the derivatives by their first order explicit Euler schemes, that is:

$$\omega'_i(t) \text{ is replaced by } \frac{\omega_i(t+\partial) - \omega_i(t)}{\partial}\,. \tag{5}$$

In (4), integer n must become large enough so that the whole time line $\mathbb{R}_{\geq 0}$ is covered—infinite integers must be used, which is also formalized in Nonstandard Analysis. Using the *shift* operators

$$^{\bullet}x(t) =_{\text{def}} x(t-\partial) \quad \text{and} \quad x^{\bullet}(t) =_{\text{def}} x(t+\partial)\,, \tag{6}$$

we can rewrite System (1) as follows:

$$\left\{\begin{array}{lll} & \frac{\omega_1^{\bullet}-\omega_1}{\partial} = f_1(\omega_1, \tau_1) & (e_1) \\ & \frac{\omega_2^{\bullet}-\omega_2}{\partial} = f_2(\omega_2, \tau_2) & (e_2) \\ \texttt{if } \gamma \ \texttt{ then } & \omega_1 - \omega_2 = 0 & (e_3) \\ \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \\ \texttt{if not } \gamma \ \texttt{ then } & \tau_1 = 0 & (e_5) \\ \texttt{and} & \tau_2 = 0 & (e_6) \end{array}\right. \tag{7}$$

Index reduction for the engaged mode can now be performed by shifting (e_3) and adding it to (7), shown in red:

$$\left\{\begin{array}{lll} & \frac{\omega_1^{\bullet}-\omega_1}{\partial} = f_1(\omega_1, \tau_1) & (e_1) \\ & \frac{\omega_2^{\bullet}-\omega_2}{\partial} = f_2(\omega_2, \tau_2) & (e_2) \\ \texttt{if } \gamma \ \texttt{ then } & \omega_1 - \omega_2 = 0 & (e_3) \\ \texttt{and} & \color{red}{\omega_1^{\bullet} - \omega_2^{\bullet} = 0} & \color{red}{(e_3^{\bullet})} \\ \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \\ \texttt{if not } \gamma \ \texttt{ then } & \tau_1 = 0 & (e_5) \\ \texttt{and} & \tau_2 = 0 & (e_6) \end{array}\right. \tag{8}$$

System (8) will be used to generate restart equations at impulsive mode change $\gamma : \text{F} \to \text{T}$. At the considered instant, we have ${}^\bullet\gamma = \text{F}$ and $\gamma = \text{T}$, see (6) for the definition of shift operators. Unfolding System (8) at the two successive previous (shown in blue) and current (shown in black) instants yields, by taking the actual values for the guard at those instants into account:

$$\begin{array}{ll} & \left\{\begin{array}{ll} \frac{\omega_1 - {}^\bullet\omega_1}{\partial} = f_1({}^\bullet\omega_1, {}^\bullet\tau_1) & ({}^\bullet e_1^\partial) \\ \frac{\omega_2 - {}^\bullet\omega_2}{\partial} = f_2({}^\bullet\omega_2, {}^\bullet\tau_2) & ({}^\bullet e_2^\partial) \\ {}^\bullet\tau_1 = 0 & \\ {}^\bullet\tau_2 = 0 & \\ \frac{\omega_1^\bullet - \omega_1}{\partial} = f_1(\omega_1, \tau_1) & \\ \frac{\omega_2^\bullet - \omega_2}{\partial} = f_2(\omega_2, \tau_2) & \\ \omega_1 - \omega_2 = 0 & (e_3) \\ \omega_1^\bullet - \omega_2^\bullet = 0 & \\ \tau_1 + \tau_2 = 0 & \end{array}\right. \begin{array}{l} \text{previous} \\ \\ \\ \\ \\ \\ \text{current} \\ \\ \\ \end{array} \end{array} \tag{9}$$

We regard System (9) as an algebraic system of equations with dependent variables being the leading variables of System (8) at the previous and current instants: ${}^\bullet\tau_i, {}^\bullet\omega_i; \tau_i, \omega_i^\bullet$ for $i = 1, 2$. System (9) is singular since the following subsystem possesses five equations and only four dependent variables $\omega_1, \omega_2, {}^\bullet\tau_1, {}^\bullet\tau_2$:

$$\left\{\begin{array}{ll} \frac{\omega_1 - {}^\bullet\omega_1}{\partial} = f_1({}^\bullet\omega_1, {}^\bullet\tau_1) & ({}^\bullet e_1^\partial) \\ \frac{\omega_2 - {}^\bullet\omega_2}{\partial} = f_2({}^\bullet\omega_2, {}^\bullet\tau_2) & ({}^\bullet e_2^\partial) \\ {}^\bullet\tau_1 = 0 & \\ {}^\bullet\tau_2 = 0 & \\ \omega_1 - \omega_2 = 0 & (e_3) \end{array}\right. \tag{10}$$

We propose to resolve the conflict in (10) by applying the following causality principle:

Principle 1 (causality) *What was done at the previous instant cannot be undone at the current instant.*

This leads to removing, from subsystem (10), the conflicting equation (e_3), thus getting the following problem for computing restart values at mode change $\gamma : \text{F} \to \text{T}$:

$$\text{solve for } \omega_1^\bullet, \omega_2^\bullet, \tau_1, \tau_2 \text{ the following system, and then, use } \omega_1^\bullet, \omega_2^\bullet \text{ as restart values for the velocities:} \quad \left\{\begin{array}{l} \omega_1, \omega_2, {}^\bullet\tau_1, {}^\bullet\tau_2 \text{ set by previous instant} \\ \frac{\omega_1^\bullet - \omega_1}{\partial} = f_1(\omega_1, \tau_1) \\ \frac{\omega_2^\bullet - \omega_2}{\partial} = f_2(\omega_2, \tau_2) \\ \omega_1^\bullet - \omega_2^\bullet = 0 \\ \tau_1 + \tau_2 = 0 \end{array}\right. \tag{11}$$

Note that the consistency equation $(e_3) : \omega_1 - \omega_2 = 0$ has been removed from System (11), thus modifying the original model. However, this removal occurs only at mode change events $\gamma : \text{F} \to \text{T}$. What we have done amounts to *delaying by one infinitesimal time step the satisfaction of some of the constraints in force in the new mode* $\gamma = \text{T}$. Since our time step is infinitesimal, this takes zero real time. The move from System (9) to System (11) must be automatized, hence we need:

Basic tool 2 *We need a graph based algorithm for identifying the conflicting equations possibly occurring at a mode change. This is addressed in Section 3.2.3 of this report, where the Dulmage-Mendelsohn decomposition of a bipartite graph is presented.*

Now, System (11) is not effective yet, since it involves, in the denominator of the left-hand side of the first two equations, the infinitesimal number ∂. Letting $\partial \searrow 0$ in System (11) requires a different kind of technique, not related to structural analysis, to identify impulsive variables (here, the two torques) and to handle them properly. Details for these techniques are found in Sections 2.5.2 and 10 of [3].

3 Structural vs. numerical analysis of systems of equations

In this section, we develop the structural analysis for a system of algebraic equations, i.e., equations in a Euclidian space, with no dynamics (no time, no derivative), of the form

$$f_i(y_1,\ldots,y_k,x_1,\ldots,x_n)=0,\quad i=1,\ldots,m\,. \tag{12}$$

This system is rewritten as $F(Y,X)=0$, where Y and X denote the vectors $(y_1,\ldots,y_k)$ and $(x_1,\ldots,x_n)$, respectively, and F is the vector $(f_1,\ldots,f_m)$. This system has m equations, k free variables (whose values are given) collected in vector Y, and n *dependent variables* (or unknowns) collected in vector X. Throughout this section , we assume that the f_i's are all of class $\mathcal{C}^1$ at least.

If the considered system (12) is square, i.e., if $m=n$, the *Implicit Function Theorem* (see, e.g., Theorem 10.2.2 in [10]) states that, if $(\nu_Y,\nu_X)\in\mathbb{R}^{k+n}$ is a value for the pair (Y,X) such that $F(\nu_Y,\nu_X)=0$ and the Jacobian matrix of F with respect to X evaluated at (ν_Y,ν_X) is nonsingular, then there exists, in an open neighborhood U of ν_Y, a unique vector of functions G such that $F(v,G(v))=0$ for all $v\in U$. In words, Eq. (12) uniquely determines X as a function of Y, locally around ν_Y.

Denote by $\mathbf{J}_XF$ the above-mentioned Jacobian matrix. Solving $F=0$ for X, given a value ν_Y for Y, requires forming $\mathbf{J}_XF(\nu_Y)$ as well as inverting it. One could avoid inverting $\mathbf{J}_XF$ (or even considering it at all), by focusing instead on its *structural nonsingularity,* introduced in Section 3.1.

3.1 Structural nonsingularity of square matrices

Say that a subset of a Euclidian space is *exceptional* if it has zero Lebesgue measure. Say that a property $P(x_1,\ldots,x_k)$ involving the real variables $x_1,\ldots,x_k$ holds *almost everywhere* if it holds for every $x_1,\ldots,x_k$ outside an exceptional subset of $\mathbb{R}^k$.

Square matrix P of size n is a *permutation matrix* if and only if there exists a permutation σ of the set $\{1,\ldots,n\}$ such that $p_{ij}=1$ if $j=\sigma(i)$ and $p_{ij}=0$ otherwise. Pre-multiplication (respectively, post-multiplication) of a matrix A by a permutation matrix results in permuting the rows (resp., the columns) of A.

These preliminary definitions and properties make it possible to prove the following lemma. This result, dealing in particular with the invertibility of sparse matrices, is part of the folklore of High Performance Computing. However, we were unable to find any proper reference in which it is clearly stated (let alone proven).

Lemma 1 *Let A be an $n\times m$-matrix with $m\geq n$. The following two properties are equivalent:*

1. *There exist two permutation matrices P and Q, such that $PAQ=[\,B_1\ B_2\,]$, where B_1 is square with non-zero entries on its main diagonal—we say that A is* structurally onto*;*
2. *Matrix A remains almost everywhere onto (surjective) when its non-zero entries vary over some neighborhood.*

Proof We consider the linear equation $AX=Y$, where $Y\in\mathbb{R}^n$ has entries $y_1,\ldots,y_n$, and X is the unknown vector with real entries $x_1,\ldots,x_m$. With a suitable renumbering of the coordinates of X, we can assume that Q is the identity matrix. For this proof, we will need to formalize what we mean by

$$\text{“letting the non-zero entries of matrix } A \text{ vary over a neighborhood.”} \tag{13}$$

To this end, we consider the *non-zero pattern* of matrix A, i.e., the subset $\mathcal{P}\subseteq[1,n]\times[1,m]$ collecting the pairs (i,j) such that a_{ij} is non-zero; let K be the cardinal of $\mathcal{P}$. Order the set $\mathcal{P}$ by lexicographic order, namely $(i,j)<(i',j')$ iff either $i<i'$, or $i=i'\ \wedge\ j<j'$. Doing this defines an order preserving bijection $\psi:[1,K]\to\mathcal{P}$. The non-zero entries of matrix A are then collected in

the vector $\mathcal{A}$ of $\mathbb{R}^K$ whose components are the $a_{\psi(k)}, k = 1, \ldots, K$. To every neighborhood U of $\mathcal{A}$, we can thus associate the set of $(n\times m)$-matrices

$$V \;=\; \left\{A' = (a'_{ij}) \,\middle|\, a'_{ij} = \xi_{\psi^{-1}(i,j)} \text{ if } (i,j) \in \mathcal{P},\ 0 \text{ otherwise} \right\}, \tag{14}$$

where the vector of $\mathbb{R}^K$, whose entries are the ξ_k for $1{\leq}k{\leq}K$, ranges over U. A set V of matrices obtained in this way is called a *zero-pattern preserving neighborhood of A,* or simply *neighborhood of A* when "zero-pattern preserving" is understood from the context. With this preliminary, we successively prove the two implications.

1 $\Longrightarrow$ 2: We need to prove the existence of a zero-pattern preserving neighborhood of A such that, for every Y, there exists an X such that $A'X = Y$ holds almost everywhere when matrix A' ranges over this neighborhood. With a suitable renumbering of the equations, we can assume that P is the identity matrix.

Let V_1 be a zero-pattern preserving neighborhood of A such that the first diagonal term a'_{11} of $A' = PA'$ remains non-zero when A' varies over V_1. Since $a'_{11} \neq 0$ holds when A' varies over V_1, one can perform, on A', the first step of a Gaussian elimination by using a'_{11} as a pivot. As x_1 is thus expressed in terms of $x_2, \ldots, x_m, y_1, \ldots, y_n$, the first equation may then be removed. This yields a reduced equation $A'_1X_1 = Y_1$, where A'_1 is an $(n-1)\times(m-1)$-matrix, $Y_1 \in \mathbb{R}^{n-1}$ collects $y_{\sigma(2)}, \ldots, y_{\sigma(n)}$, and $X_1 \in \mathbb{R}^{m-1}$ collects the unknowns $x_2, \ldots, x_m$. Matrix A'_1 has diagonal entries equal to $a^{1\prime}_{kk} = a'_{kk} - a'_{k1}/a'_{11}$, for $k = 2, \ldots, n$. Since a'_{11} is non-zero, the set of entries a'_{ij} of matrix A' causing $a^{1\prime}_{kk} = 0$ for some k is exceptional in $\mathbb{R}^{n\times m}$ since it requires the condition $a'_{k1} = a'_{kk}a'_{11}$ to hold. The corresponding set of entries a'_{ij} of matrix A' is exceptional as well; we denote by Ξ_1 this exceptional subset of $\mathbb{R}^{n\times m}$. Thus, we assume that A' stays within $V_1 \setminus \Xi_1$.

This ensures that the diagonal entries of matrix A' are all non-zero. Thus, we can express x_2 in terms of $x_3, \ldots, x_m, y_{\sigma(2)}$. We are then left with a further reduced equation $A'_1X'_1 = Y'_1$ for which the same argument applies, namely: if matrix A' stays within a certain zero-pattern preserving neighborhood V_2 of A, but does not belong to some exceptional set Ξ_2, then the diagonal entries of A'_1 are all non-zero. And so on, by reducing the number of equations and unknowns.

By iterating this process, we prove the existence of sets of matrices $V_1, \ldots, V_n$ and $\Xi_1, \Xi_2, \ldots, \Xi_n$ such that, for every matrix belonging to $(V_1\cap\cdots\cap V_n)\setminus(\Xi_1\cup\Xi_2\cup\cdots\cup\Xi_n)$, the values of $x_1, x_2, \ldots, x_n$ are uniquely determined as functions of the coordinates of Y and the other variables $x_{n+1}, \ldots, x_m$, by pivoting. This concludes the proof of this implication, since $V_1 \cap \cdots \cap V_n$ is a zero-pattern preserving neighborhood of A and set $\Xi_1 \cup \Xi_2 \cup \cdots \cup \Xi_n$ is exceptional.

2 $\Longrightarrow$ 1: We assume the existence of a zero-pattern preserving neighborhood V of A and an exceptional set Ξ such that A' remains onto when varying over $V \setminus \Xi$. For each onto A', there exists at least one permutation matrix Q such that

$$A'Q \;=\; \left[\begin{array}{cc} B_1 & B_2 \end{array}\right] \tag{15}$$

where B_1 is a square invertible matrix. Neighborhood V decomposes as the union $V = \bigcup_\sigma V_\sigma$, where V_σ collects the matrices A' for which decomposition (15) holds with the matrix Q representing permutation σ. Since the number of permutations is finite, at least one of the V_σ has non-empty interior, say, for $\sigma = \sigma_*$. We can thus trim V to V_{σ_*}, which still is a zero-pattern preserving neighborhood of A, and we call it again V for convenience.

Hence, there exists a permutation matrix Q such that decomposition (15) holds for every $A' \in V$. By Condition 2 of Lemma 1, the pre-image of 0 by the map $V \ni A' \mapsto \det(B_1)$, where $\det(M)$ denotes the determinant of a square matrix M, is an exceptional set. Now, we have $\det(B_1) = \sum_{i=1}^{n}(-1)^{i+1}b_{i1}\det(B_{i1})$, where B_{ij} is the submatrix of B_1 obtained by erasing row i and column j. Since B_1 is almost everywhere invertible, there must be some $i \in \{1, \ldots, n\}$ such that $b_{i1} \neq 0$ and B_{i1} is almost everywhere invertible—otherwise we would have $\det(B_1) = 0$ for any $A' \in W$, where W is some neighborhood contained in V. Let P^1 be the permutation matrix exchanging rows 1 and i, so we replace B_1 by $B^1_1 = P^1B_1$. Now, since we have $b^1_{11} = b_{i1} \neq 0$ and B^1_{11} is almost everywhere invertible. This process is iterated $m-1$ times. For the last 1-row matrix,

its diagonal entry is non-zero, otherwise the determinant of the corresponding B_1 matrix would be zero in a neighborhood of A. The wanted left permutation matrix is $P = P^{m-1}...P^2P^1$. □

Lemma 1 specializes to the following simpler result:

Corollary 2 *Let A be an $n \times n$-matrix. The following two properties are equivalent:*

1. *There exist two permutation matrices P and Q, such that $PAQ = B$, where B has non-zero entries on its main diagonal—we say that A is* structurally nonsingular*;*
2. *Matrix A remains almost everywhere invertible when its non-zero entries vary over some neighborhood.*

3.2 Structural analysis of algebraic equations

In this section, we focus on systems of algebraic equations $F(Y, X) = 0$ of the form (12), that is, equations involving no time and no dynamics. We will need to handle variables, their valuations, and vectors thereof. To this end, the following conventions will be used (unless no confusion occurs from using more straightforward notations):

Notations 3 Lowercase letters $(x, y, ...)$ denote scalar real variables; capitals $(X, Y, ...)$ denote vectors of real variables; whenever convenient, we regard X as a set of variables and write $x \in X$ to refer to an entry of X. We adopt similar conventions for functions $(f, g,...)$ and vectors of functions $(F, G,...)$. A value for a variable x is generically denoted by ν_x, and similarly for X and ν_X. □

As explained in the introduction of this section, the existence and uniqueness of solutions for system (12) relates to the invertibility of its Jacobian $\mathbf{J}_X F$. The structural analysis of system (12) is built on top of the structural nonsingularity of its Jacobian, introduced in Section 3.1. Its full development requires some background on graphs.

3.2.1 Background on graphs:

This short background is compiled from [5, 12, 13]. Given a graph $\mathcal{G} = (V, E)$, where V and $E \subseteq V \times V$ are the sets of vertices and edges, a *matching* $\mathcal{M}$ of $\mathcal{G}$ is a set of edges of $\mathcal{G}$ such that no two edges in $\mathcal{M}$ are incident on a common vertex. Matching $\mathcal{M}$ is called *complete* if it covers all the vertices of $\mathcal{G}$. An *$\mathcal{M}$-alternating path* is a path of $\mathcal{G}$ whose edges are alternatively in $\mathcal{M}$ and not in $\mathcal{M}$—we simply say "alternating path" when $\mathcal{M}$ is understood from the context. A vertex is *matched* in $\mathcal{M}$ if it is an endpoint of an edge in $\mathcal{M}$, and *unmatched* otherwise.

A *bipartite graph* is a graph $\mathcal{G} = (L \cup R, E)$, where $E \subseteq L \times R$. Let $F(Y, X) = 0$ be a system of equations of the form (12); its *bipartite graph* $\mathcal{G}_F$ is the graph having $F \cup X$ as its set of vertices and an edge (f_i, x_j) if and only if variable x_j occurs in function f_i.

3.2.2 Square systems

We first consider square systems, in which the equations and the dependent variables are in equal numbers.

Definition 4 (structural nonsingularity) *System $F(Y, X) = 0$ is called* structurally nonsingular *if its bipartite graph $\mathcal{G}_F$ possesses a complete matching.*

A structurally nonsingular system is necessarily square, i.e., with equations and dependent variables in equal numbers. Note that Condition 1 of Corollary 2 is a matrix reformulation of the graph theoretic condition stated in Definition 4. Thus, F is structurally nonsingular in the sense of Definition 4, if and only it its Jacobian $\mathbf{J}_X F$, evaluated at any pair (ν_Y, ν_X) such that $F(\nu_Y, \nu_X) = 0$, is structurally

nonsingular in the sense of Corollary 2. The link to numerical regularity is thus formalized in the following lemma:[1]

Lemma 5 *Assume that F is of class $\mathcal{C}^1$. The following properties are equivalent:*

1. *System $F(Y,X)=0$ is structurally nonsingular;*
2. *For every (ν_Y,ν_X) satisfying $F(\nu_Y,\nu_X)=0$, the Jacobian matrix $\mathbf{J}_X F(\nu_Y,\nu_X)$ remains generically[2] nonsingular when its non-zero coefficients vary over some neighborhood.*

Property 2 is interpreted using the argument developed in (13,14): every $n\times n$-matrix having the same non-zero pattern as $\mathbf{J}_X F(\nu_Y,\nu_X)$ identifies with a unique vector of $\mathbb{R}^K$, where K is the cardinal of this non-zero pattern. We denote by $\mathcal{J}_X F\in\mathbb{R}^K$ the image of the Jacobian $\mathbf{J}_X F(\nu_Y,\nu_X)$ obtained via this correspondence. Property 2 says:

$$\begin{array}{l}\text{There exist an open neighborhood } U \text{ of } \mathcal{J}_X F \text{ in } \mathbb{R}^K, \text{ and a subset}\\ V\subseteq U \text{ such that: } (i) \text{ the set } U\setminus V \text{ has zero Lebesgue measure, and}\\ (ii) \text{ every } \mathcal{J}\in V \text{ yields a regular matrix.}\end{array} \tag{16}$$

In the sequel, the so defined sets U and V will be denoted by

$$U_F(\nu_Y,\nu_X) \quad\text{and}\quad V_F(\nu_Y,\nu_X) \tag{17}$$

or simply U_F and V_F when no confusion can result. □

Let $F(Y,X)=0$ be structurally nonsingular and let $\mathcal{M}$ be a complete matching for it. Using $\mathcal{M}$, graph $\mathcal{G}_F$ can be directed as follows: edge (f_i,x_j) is directed from f_i to x_j if $(f_i,x_j)\in\mathcal{M}$, from x_j to f_i otherwise. Denote by $\vec{\mathcal{G}}_F^{\mathcal{M}}$ the resulting directed graph. The following result holds ([11], Chapter 6.10):

Lemma 6 *The strongly connected components of $\vec{\mathcal{G}}_F^{\mathcal{M}}$ are independent of $\mathcal{M}$.*

Each strongly connected component defines a *block* of F, consisting of the set of all equations that are vertices of this component. Blocks are partially ordered and we denote by $\preceq_F$ this order. Extending $\preceq_F$ to a total order (via topological sorting) yields an ordering of equations that puts the Jacobian matrix $\mathbf{J}_X F$ in *Block Triangular Form* (BTF).

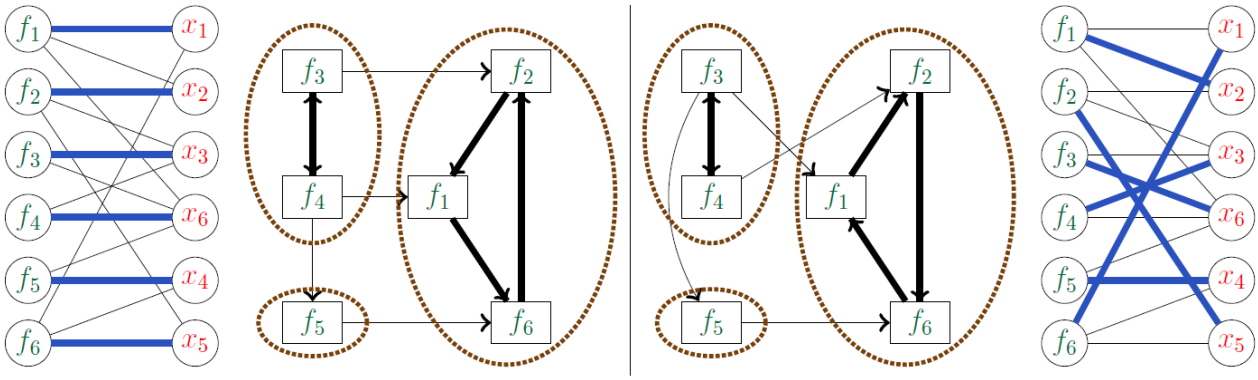


Figure 2: Illustrating Lemma 6 (we only show the projection of strongly connected components on function nodes).

This lemma is illustrated in Figure 2, inspired by lecture notes by J-Y. L'Excellent and Bora Uçar, 2010. In this figure, a same bipartite graph $\mathcal{G}$ is shown twice (left and right), with the

[1] This result is implicitly used in [15, 14, 16, 17], as well as in a major part of the literature on the structural analysis of DAE systems.

[2] Generically means: outside a set of values of Lebesgue measure zero.

equations sitting on the left-hand side in green, and the variables on the right-hand side in red. Two complete matchings $\mathcal{M}_1$ (left) and $\mathcal{M}_2$ (right) are shown in thick blue, with other edges of the bipartite graph being black. The restriction, to the equation vertices, of the two directed graphs $\vec{\mathcal{G}}_F^{\mathcal{M}_1}$ (left) and $\vec{\mathcal{G}}_F^{\mathcal{M}_2}$ (right) are shown on both sides. Although the two directed graphs $\vec{\mathcal{G}}_F^{\mathcal{M}_1}$ and $\vec{\mathcal{G}}_F^{\mathcal{M}_2}$ differ, the resulting block structures (encircled in dashed brown) are identical.

3.2.3 Non-square systems, Dulmage-Mendelsohn decomposition

In our development, we will also encounter non-square systems of algebraic equations. For general systems (with a number of variables not necessarily equal to the number of equations, $n \neq m$ in (12)), call *block* a pair $\beta = (\mathbf{f}, \mathbf{x})$, consisting of a subsystem $\mathbf{f}=0$ of system $F=0$ and the subset $\mathbf{x}$ of its dependent variables.

Definition 7 (Dulmage-Mendelsohn) *For $F = 0$ a general system of algebraic equations, the* Dulmage-Mendelsohn decomposition [16] *of $\mathcal{G}_F$ yields the partition of system $F = 0$ into the following three blocks, given some matching of maximal cardinality for $\mathcal{G}_F$:*

- *block $\beta_{\mathfrak{O}}$ collects the variables and equations reachable via some alternating path from some unmatched equation;*
- *block $\beta_{\mathfrak{U}}$ collects the variables and equations reachable via some alternating path from some unmatched variable;*
- *block $\beta_{\mathfrak{E}}$ collects the remaining variables and equations.*

Blocks $\beta_{\mathfrak{O}}, \beta_{\mathfrak{U}}, \beta_{\mathfrak{E}}$ are the $\mathfrak{O}$verdetermined, $\mathfrak{U}$nderdetermined, and $\mathfrak{E}$nabled parts of F.

Statement 1 of the following lemma ensures that Definition 7 is meaningful:

Lemma 8

1. *The triple $(\beta_{\mathfrak{U}}, \beta_{\mathfrak{E}}, \beta_{\mathfrak{O}})$ defined by the Dulmage-Mendelsohn decomposition does not depend on the particular maximal matching used for its definition. We thus write*

$$\mathrm{DM}(F) \quad = \quad (\beta_{\mathfrak{U}}, \beta_{\mathfrak{E}}, \beta_{\mathfrak{O}})\,. \tag{18}$$

2. *System F is structurally nonsingular if and only if the overdetermined and underdetermined blocks $\beta_{\mathfrak{O}}$ and $\beta_{\mathfrak{U}}$ are both empty.*

A refined DM decomposition exists, which in addition refines block $\beta_{\mathfrak{E}}$ into its indecomposable block triangular form. This is a direct consequence of applying Lemma 6 to $\beta_{\mathfrak{E}}$ in order to get its BTF. The following lemma is useful in the structural analysis of multimode DAE systems [2, 3]:

Lemma 9 *Let $\mathrm{DM}(F) = (\beta_{\mathfrak{U}}, \beta_{\mathfrak{E}}, \beta_{\mathfrak{O}})$ be the Dulmage-Mendelsohn decomposition of $F = 0$. Then, no overdetermined block exists in the Dulmage-Mendelsohn decomposition of $F \setminus \beta_{\mathfrak{O}}$.*

Comment 10 We could refine this result by only removing unmatched equations with reference to the matching of maximal cardinality used for generating $\mathrm{DM}(F)$. Then, denoting by F' the remaining subsystem of F, we would still get that no overdetermined block exists in the Dulmage-Mendelsohn decomposition of F', and $F' \supseteq (F \setminus \beta_{\mathfrak{O}})$ (the inclusion being strict in general). However, this policy depends on the particular matching. In contrast, the rule defined by Lemma 9 is independent from the matching. This Lemma is used in [2, 3] to generate restart code at mode changes, see Section 5.3.2 of this report; the independence with respect to the particular matching considered was crucial to ensure the determinism of code generation.

3.3 Practical use of structural analysis

We distinguish between two different uses of structural analysis. The classical use assumes that we already have a close guess of a solution, we call it the *local use.* By contrast, the other use is called *global.* We explain both here.

3.3.1 Local use of structural analysis

If a square matrix is structurally singular, then it is singular. The converse is false: structural nonsingularity does not guarantee nonsingularity. Therefore, the practical use of Definition 4 is as follows: We first check if $F = 0$ is structurally nonsingular. If not, then we abort searching for a solution. Otherwise, we then proceed to computing a solution, which may or may not succeed depending on the actual numerical regularity of the Jacobian matrix $\mathbf{J}_X F$.

Structural analysis relies on the Implicit Function Theorem for its justification, as stated at the beginning of this section. Therefore, the classical use of structural analysis is local: let $F(Y, X) = 0$ be a structurally nonsingular system of equations with dependent variables X, and let ν_X satisfy $F(\nu_Y, \nu_X) = 0$ for a given value ν_Y for Y; then, the system remains (generically) nonsingular if ν varies in a neighborhood of ν_Y. This is the situation encountered in the running of ODE or DAE solvers, since a close guess of the solution is known from the previous time step.

3.3.2 Non-local use of structural analysis

We are, however, also interested in invoking the structural nonsingularity of system $F(Y, X) = 0$ when no close guess is known. This is the situation encountered when handling mode changes of multimode DAE systems [2, 3]. We thus need another argument to justify the use of structural analysis in this case.

As a prerequisite, we recall basic definitions regarding smooth manifolds (see, e.g., [8], Section 4.7). In the next lemma, we consider the system $F(Y, X) = 0$ for a fixed value of Y, thus Y is omitted.

Lemma 11 *Let $k, m, n \in \mathbb{N}$ be such that $m < n$ and set $p = n - m$. For $\mathcal{S} \subset \mathbb{R}^n$ and $x^* = (x_1^*, \ldots, x_n^*) \in \mathcal{S}$, the following properties are equivalent:*

1. *There exists an open neighborhood V of x^* and a $\mathcal{C}^k$-diffeomorphism $F : V \to W \subset \mathbb{R}^n$ such that $F(x^*) = 0$ and $F(\mathcal{S} \cap V)$ is the intersection of W with the subspace of $\mathbb{R}^n$ defined by the m equations $w_{p+1} = 0, \ldots, w_n = 0$, where w_i denote the coordinates of points belonging to W.*

2. *There exists an open neighborhood V of x^*, an open neighborhood U of 0 in $\mathbb{R}^p$ and a homeomorphism $\psi : U \to \mathcal{S} \cap V$, such that ψ, seen as a map with values in $\mathbb{R}^n$, is of class $\mathcal{C}^k$ and has rank p at 0—i.e., $\psi'(0)$ has rank p.*

Statement 1 means that $\mathcal{S}$ is, in a neighborhood of x^*, the solution set of the system of equations $f_1(X) = 0, \ldots, f_m(X) = 0$ in the n-tuple of dependent variables X, where $F = (f_1, \ldots, f_m)$. Statement 2 expresses that ψ is a parameterization of $\mathcal{S}$ in a neighborhood of x^*, of class $\mathcal{C}^k$ and rank p—meaning that $\mathcal{S}$ has dimension p in a neighborhood of x^*.

In order to apply Lemma 11 to the non-local use of structural analysis, we have to study the case of square systems of equations (i.e., let $m = n$). Consider a system $F(X) = 0$, where $X = (x_1, \ldots, x_n)$ is the n-tuple of dependent variables and $F = (f_1, \ldots, f_n)$ is an n-tuple of functions $\mathbb{R}^n \to \mathbb{R}$ of class $\mathcal{C}^k$. Let $\mathcal{S} \subseteq \mathbb{R}^n$ be the solution set of this system, that we assume is non-empty. To be able to apply Lemma 11, we augment X with one extra variable z, i.e., we set $Z =_{\text{def}} X \cup \{z\}$ and we now regard our formerly square system $F(X) = 0$ as an augmented system $G(Z) = 0$ where $G(X, z) =_{\text{def}} F(X)$. Extended system G possesses n equations and $n+1$ dependent variables, hence, $p = 1$, and its solution set is equal to $\mathcal{S} \times \mathbb{R}$. Let $x^* \in \mathcal{S}$ be a solution of $F = 0$. Then, we have $G(x^*, z^*) = 0$ for any $z^* \in \mathbb{R}$. Assume further that the Jacobian matrix $\mathbf{J}_X F(x)$ is nonsingular at x^*. Then, it remains nonsingular in a neighborhood of x^*. Hence, the Jacobian matrix $\mathbf{J}_X G(x, z)$ has rank n in a neighborhood V of (x^*, z^*) in $\mathbb{R}^{n+1}$. We can thus extend G by adding one more function in such a way that the resulting $(n+1)$-tuple $\bar{G}$ is a $\mathcal{C}^k$-diffeomorphism, from V to an open set W of $\mathbb{R}^{n+1}$. The extended system $\bar{G} = 0$ satisfies Property 1 of Lemma 11. By Property 2 of Lemma 11, $(\mathcal{S} \times \mathbb{R}) \cap V$ has dimension 1, implying that $\mathcal{S} \cap \mathbf{proj}_{\mathbb{R}^n}(V)$ has dimension < 1. This analysis is summarized in the following lemma:

Lemma 12 *Consider the square system $F(X) = 0$, where $X = (x_1, \ldots, x_n)$ is the n-tuple of dependent variables and $F = (f_1, \ldots, f_n)$ is an n-tuple of functions $\mathbb{R}^n \to \mathbb{R}$ of class $\mathcal{C}^k, k \geq 1$. Let $\mathcal{S} \subseteq \mathbb{R}^n$ be the solution set of this system. Assume that system $F = 0$ has solutions and let x^* be such a solution. Assume further that the Jacobian matrix $\mathbf{J}_X F(x)$ is nonsingular at x^*. Then, there exists an open neighborhood V of x^* in $\mathbb{R}^n$, such that $\mathcal{S} \cap V$ has dimension < 1.*

The implications for the non-local use of structural analysis are as follows. Consider the square system $F(X) = 0$, where $X = (x_1, \ldots, x_n)$ is the n-tuple of dependent variables and $F = (f_1, \ldots, f_n)$ is an n-tuple of functions $\mathbb{R}^n \to \mathbb{R}$ of class $\mathcal{C}^k$. We first check the structural nonsingularity of $F = 0$. If structural nonsingularity does not hold, then we abort solving the system. Otherwise, we proceed to solving the system and the following cases can occur:

1. the system possesses no solution;
2. its solution set is nonempty, of dimension < 1; or
3. its solution set is nonempty, of dimension ≥ 1.

Based on Lemma 12, one among cases 1 or 2 will hold, generically, whereas case 3 will hold exceptionally. Thus, structurally, we know that the system is not underdetermined. This is our justification of the non-local use of structural analysis. Note that the existence of a solution is not guaranteed. Furthermore, unlike for the local use, uniqueness is not guaranteed either. Lemma 12 only states that there cannot be two arbitrarily close solutions of $F = 0$, as its conclusion $\mathcal{S} \cap V = \{x^*\}$ involves an open neighborhood V of x^*. Of course, subclasses of systems for which existence and uniqueness are guaranteed are of interest.

3.4 Equations with existential quantifiers

To support the structural analysis for the method of *differential arrays* advocated by Campbell and Gear [7] (see also Section 4.3 of this report), we will need to develop a structural analysis for the following class of (possibly non-square) algebraic systems of equations with existential quantifier:[3]

$$\exists W : F(X, W, Y) = 0 \ , \tag{19}$$

where (X, W) collects the dependent variables of system $F(X, W, Y) = 0$. Y and X are as before, and the additional tuple W collects supplementary variables, of no interest to us, whence their elimination by existential quantification. Our aim is to find structural conditions ensuring that (19) defines a partial function $Y \to X$, meaning that the value of tuple X is uniquely defined by the satisfaction of (19), given a value for tuple Y.

At this point, a fundamental difficulty arises. Whereas (19) is well defined as an abstract relation, it cannot in general be represented by a projected system of smooth algebraic equations of the form $G(X, Y) = 0$. Such a G can be associated to (19) only in subclasses of systems.[4] In general, no extension of the Implicit Function Theorem exists for systems of the form (19); thus, one cannot apply as such the arguments developed in Section 3.3.2. Therefore, we will reformulate our requirement differently. Say that

$$\begin{array}{l} \nu_Y \text{ is } \textit{consistent} \text{ if the system of equations } F(X, W, \nu_Y) = 0 \\ \text{possesses a solution for } (X, W). \end{array} \tag{20}$$

Consider the following property for the systen $F(X, W, Y) = 0$: for every consistent ν_Y,

$$\left.\begin{array}{l} F(\nu_X^1, \nu_W^1, \nu_Y) = 0 \\ F(\nu_X^2, \nu_W^2, \nu_Y) = 0 \end{array}\right\} \implies \nu_X^1 = \nu_X^2 \ , \tag{21}$$

[3] To our knowledge, the results of this section are not part of the folklore of sparse matrix algebra.

[4] Examples are linear systems for which elimination is easy, and polynomial systems for which it is doable—but expensive—using Gröbner bases.

expressing that X is independent of W given a consistent tuple of values for Y. To find structural criteria guaranteeing (21), we will consider the algebraic system $F(X, W, Y) = 0$ with X, W, Y as dependent variables (i.e., values for the entries of Y are no longer seen as being given).

Definition 13 *System* (19) *is* structurally nonsingular *if the following two conditions hold, almost everywhere when the non-zero coefficients of the Jacobian matrix* $\mathbf{J}_{X,W,Y}F$ *vary over some neighborhood:*[5]

- *consistent values for* Y *exist;*
- *condition* (21) *holds.*

The structural nonsingularity of (19) can be checked by using the DM decomposition of F as follows. Let $(\beta_{\mathfrak{U}}, \beta_{\mathfrak{E}}, \beta_{\mathfrak{O}}) = \mathrm{DM}(F)$ be the DM decomposition of $F(X, W, Y) = 0$, with (X, W, Y) as dependent variables. We further assume that the regular block $\beta_{\mathfrak{E}}$ is expressed in its block triangular form, see Lemma 6 and comments thereafter. Let $\mathbf{B}$ be the set of all indecomposable blocks of $\beta_{\mathfrak{E}}$ and $\preceq$ be the partial order on $\mathbf{B}$ following Lemma 6. The following holds:

Lemma 14 *System* (19) *is* structurally nonsingular *if and only if:*

1. *Block* $\beta_{\mathfrak{O}}$ *is empty;*
2. *Block* $\beta_{\mathfrak{U}}$ *involves no variable belonging to* X*;*
3. *For every* $\beta \in \mathbf{B}$ *containing some variable from* X*, then,* β *contains no variable from* W*, and for every* $\beta' \prec \beta$ *and every directed edge* $(z', z) \in \vec{\mathcal{G}}_F^{\mathcal{M}}$ *such that* $z' \in \beta'$*,* $z \in \beta$*, and* $\mathcal{M}$ *is an arbitrary complete matching for* $\mathcal{G}_F$*, then* $z' \notin W$*.*

Conditions 1 and 2 speak by themselves. Regarding Condition 3, the intuition is the following. Every directed path of $\vec{\mathcal{G}}_F^{\mathcal{M}}$, originating from W and terminating in X, of minimal length, must traverse Y. Consequently, W influences X "through" Y only.

Proof We successively prove the if and then the only if parts.

"If" part: By Condition 1, there exist consistent values for Y. By condition 2, no variable of X belongs to the underdetermined part of $\mathrm{DM}(F)$. It remains to show that condition (21) holds. By condition (3), each indecomposable block $\beta \in \mathbf{B}$ involving a variable of X has the form $G(Z, U) = 0$, where

1. Z is the n-tuple of dependent variables of $G = (g_1, \ldots, g_n)$,
2. no variable of W belongs to Z, and
3. $U \subseteq X \cup Y$ is a subset of the dependent variables of the blocks β' immediately preceding β.

Since $G = 0$ is structurally regular, fixing a value for U entirely determines all the dependent variables of block β. This proves the "if" part.

"Only if" part: We prove it by contradiction. If condition 1 does not hold, then the existence of consistent values for Y is not structurally guaranteed. If condition 2 does not hold, then the variables of X that belong to $\beta_{\mathfrak{U}}$ are not determined, even for given values of W, Y. If condition 3 does not hold, then two cases can occur:

- Some $x \in X$ and $w \in W$ are involved in a same indecomposable block $\beta \in \mathbf{B}$. Then, condition (21) will not hold for x due to the structural coupling with w through block β.
- There exists an indecomposable block $\beta \in \mathbf{B}$ of the form $G(Z, U) = 0$, where

[5] The precise meaning of this statement is the same as in Lemma 5, see the discussion therafter.

1. Z is the n-tuple of dependent variables of $G = (g_1, \ldots, g_n)$, and Z contains a variable $x \in X$,
2. no variable of W belongs to Z, and
3. some variable $w \in W$ belongs to U, the set of the dependent variables of the blocks β' immediately preceding β.

Hence, w influences x, structurally, which again violates (21). This finishes the proof. □

Algorithm 1 ExistQuantifEqn

Require: F; **return** $(b_{\mathfrak{O}}, b_{\mathfrak{U}}, F_\Sigma, \overline{F}_\Sigma)$

```
(β_𝔘, β_𝔈, β_𝔒) = DM(F)
if condition 1 of Lemma 14 holds then
    b_𝔒 ← T;
    if 2 and 3 of Lemma 14 hold then
        b_𝔘 ← T;
        partition β_𝔈 = F_Σ ∪ F̄_Σ
    else
        b_𝔘 ← F
else
    b_𝔒 ← F
```

We complement Lemma 14 with the algorithm ExistQuantifEqn (Algorithm 1), which requires a system F of the form (19). If condition 1 of Lemma 14 fails to be satisfied, then $b_{\mathfrak{O}} \leftarrow \mathrm{F}$ is returned, indicating overdetermination. If conditions 2 or 3 of Lemma 14 fail to be satisfied, then $b_{\mathfrak{U}} \leftarrow \mathrm{F}$ is returned, indicating underdetermination. Otherwise, ExistQuantifEqn succeeds and returns the value T for both Booleans, together with the decomposition $F_\Sigma \cup \overline{F}_\Sigma$ of $\beta_{\mathfrak{E}}$. In this decomposition:

- Subsystem F_Σ collects the indecomposable blocks involving variables belonging to X, so that F_Σ determines X as a function of ν_Y when ν_Y is consistent;
- Subsystem $\overline{F}_\Sigma$ collects the consistency conditions, whose dependent variables belong to $W \cup Y$.

Our background on the structural analysis of algebraic equations is now complete. In the next section, we recall the background on the structural analysis of (single-mode) DAE systems.

4 Structural analysis of DAE systems

In this section, we consider DAE systems of the form

$$f_j(x_i\text{'s and derivatives}) = 0\,, \tag{22}$$

where $x_1, \ldots, x_m$ denote the dependent signal variables (their valuations are trajectories) and $f_1 = 0, \ldots, f_n = 0$ denote the equations—with reference to notation (12), we omit the input signals collected in Y. By analogy with DAEs, we use the acronym dAE to mean *difference Algebraic Equations,* which define discrete-time dynamical systems of the form

$$f_j(x_i\text{'s and shifts}) = 0\,. \tag{23}$$

where $x_1, \ldots, x_m$ denote the dependent stream variables (their valuations are data streams, which can be regarded as real sequences), $f_1 = 0, \ldots, f_n = 0$ denote the equations, and, for $x = \{x_n \mid n = 1, 2, \ldots\}$ a stream of variables, its successive *shifts* $x^{\bullet k}$, where k is a nonnegative integer, are defined by

$$x_n^{\bullet k} =_{\text{def}} x_{n+k}\ ; \quad \text{we write for short} \quad x^\bullet =_{\text{def}} x^{\bullet 1}\,. \tag{24}$$

Also, the notation x'^k is adopted throughout this paper, instead of the more classical $x^{(k)}$, for the k-th derivative of x.

4.1 John Pryce's Σ-method for square systems

The DAE systems we consider are "square", i.e., have the form (22), with $m = n$. Call *leading variables* of System (22) the d_i-th derivatives $x_i^{\prime d_i}$ for $i = 1, \ldots, n$, where d_i is the maximal differentiation degree of variable x_i throughout $f_1 = 0, \ldots, f_n = 0$. The problem addressed by the structural analysis of DAE systems of the form (22) is the following. Regard (22) as a system of algebraic equations with the leading variables as unknowns. If this system is structurally nonsingular, then, given a value for all the $x_i^{\prime k}$ for $i = 1, \ldots, n$ and $k = 0, \ldots, d_i - 1$, a unique value for the leading variables can be computed, structurally; hence, System (22) is "like an ODE". If this is not the case, finding additional *latent equations* by differentiating suitably selected equations from (22) will bring the system to an ODE-like form, while not changing its set of solutions. Performing this is known as *index reduction.*

Algorithms were proposed in the literature for doing it efficiently. Among them, the Pantelides' algorithm [15] is the historical solution. We decided, however, to base our subsequent developments on the beautiful method proposed in [17] by J. Pryce, called the Σ-method. The *Σ-method* also covers the construction of the block triangular form and addresses numerical issues, which we do not discuss here.

Weighted bipartite graphs: We consider System (22), which is entirely characterized by its set of dependent variables X (whose generic element is denoted by x) and its set of equations $F = 0$ (whose generic element is written $f = 0$). We attach to (22) the bipartite graph $\mathcal{G} = (F \cup X, E_{\mathcal{G}})$ having an edge $(f, x) \in E_{\mathcal{G}}$ if and only if x occurs in function f, regardless of its differentiation degree. Recall that a matching is *complete* iff it involves all equations of F.

So far, $\mathcal{G}$ is agnostic with respect to differentiations. To account for this, we further equip $\mathcal{G}$ with *weights*: to each edge $(f, x) \in E_{\mathcal{G}}$ is associated a nonnegative integer d_{fx}, equal to the maximal differentiation degree of variable x in function f. This yields a *weight* for any matching $\mathcal{M}$ of $\mathcal{G}$ by the formula $w(\mathcal{M}) = \sum_{(f,x)\in\mathcal{M}} d_{fx}$. Suppose we have a solution to the following problem:

Problem 1 *Find a complete matching $\mathcal{M}$ for $\mathcal{G}$ and nonnegative integer* offsets $\{c_f \mid f \in F\}$ and $\{d_x \mid x \in X\}$, *satisfying the following conditions, where the d_{fx} are the weights as before:*

$$\begin{array}{rl} d_x - c_f \geq d_{fx} & \text{for all } (f,x) \in E_{\mathcal{G}}, \text{ with equality if } (f,x)\in\mathcal{M} \\ c_f \geq 0 & \text{for all } f \in F\,. \end{array} \tag{25}$$

Then, differentiating c_f times each function f yields a DAE system $F_\Sigma = 0$ having the following properties. Inequality $d_x \geq c_f + d_{fx}$ holds for each $x\in X$, and $d_x = c_f + d_{fx}$ holds for the unique f such that $(f, x)\in\mathcal{M}$. Hence, the leading variables of DAE system $F_\Sigma = 0$ are the d_x-th derivatives $x^{\prime d_x}$. Consequently, system $F_\Sigma = 0$, now seen as a system of algebraic equations having $x^{\prime d_x}$ as dependent variables, is structurally nonsingular by Definition 4. Hence, $F_\Sigma = 0$ is "like an ODE". The integer $k = \max_{f\in F} c_f$ is called the *index* of the system.[6]

Definition 15 *For F a DAE system, the solution to Problem 1 yields the DAE system* $F_\Sigma = \{f^{\prime c_f} \mid f\in F\}$ *together with the system of* consistency constraints $\overline{F}_\Sigma = \{f^{\prime k} \mid f\in F, 0\leq k<c_f\}$.

Knowing the offsets also allows transforming F into a system of index 1, by not performing the final round of differentiations. There are infinitely many solutions to Problem 1 with unknowns c_f and d_x, since, for example, adding the same $\ell \in \mathbb{N}_{\geq 0}$ to all c_f's and d_x's yields another solution. We thus seek for a *smallest* solution, elementwise. Hence, Problem 1 is the fundamental problem we must solve, and we seek a smallest solution for it.

The beautiful idea of J. Pryce is to propose a linear program encoding Problem 1. As a preliminary step, we claim that a bruteforce encoding of Problem 1 is the following: Find nonnegative integers

[6] We should rather say the *differentiation index* as, once again, other notions of index exist for DAEs [7] that are not relevant to our work. Also note that the standard definition of the differentiation index, used in Pryce's article [17], slightly differs from the one that we adopted in this report for the sake of clarity.

ξ_{fx}, d_x, c_f such that

$$\left.\begin{array}{r} \sum_{f:(f,x)\in E_{\mathcal{G}}} \xi_{fx} = 1 \\ \sum_{x:(f,x)\in E_{\mathcal{G}}} \xi_{fx} = 1 \\ \xi_{fx} \geq 0 \end{array}\right\} \text{complete matching}$$
$$\left.\begin{array}{r} d_x - c_f - d_{fx} \geq 0 \\ c_f \geq 0 \end{array}\right\} \text{encodes ``}\geq\text{'' in (25)} \qquad (26)$$
$$\sum_{(f,x)\in E_{\mathcal{G}}} \xi_{fx}(d_x - c_f - d_{fx}) = 0 \quad \text{encodes ``}=\text{'' in (25)}$$

where, for the formulas having no summation: x, f, and (f, x) range over X, F, and $E_{\mathcal{G}}$ respectively. We now justify our claim. Focus on the first block. Since the ξ's are nonnegative integers, they can only take values in $\{0, 1\}$ and one defines a subgraph of graph $\mathcal{G}$ by only keeping edges (f, x) such that $\xi_{fx} = 1$. The first two equations formalize that the chosen subset of edges is a matching, which is complete since all vertices of $\mathcal{G}$ are involved. The second block is a direct encoding of (25) if we ignore the additional statement "with equality iff". The latter is encoded by the last constraint (since having the sum equal to zero requires that all the terms be equal to zero). Constraint problem (26) does not account for our wish for a "smallest" solution: this will be handled separately.

Following the characterization of solutions of linear programs via *complementary slackness conditions*, every solution of problem (26) is a solution of the following dual linear programs (LP), where the ξ_{fx} and the c_f are real:

$$\begin{array}{rll} primal: & \text{maximize} & \sum_{(f,x)\in E_{\mathcal{G}}} d_{fx}\, \xi_{fx} \\ & \text{subject to} & \sum_{f:(f,x)\in E_{\mathcal{G}}} \xi_{fx} = 1 \\ & \text{and} & \sum_{x:(f,x)\in E_{\mathcal{G}}} \xi_{fx} \geq 1 \\ & \text{and} & \xi_{fx} \geq 0 \end{array} \qquad (27)$$

$$\begin{array}{rll} dual: & \text{minimize} & \sum_x d_x - \sum_f c_f \\ & \text{subject to} & d_x - c_f \geq d_{fx} \\ & \text{and} & c_f \geq 0 \end{array} \qquad (28)$$

where, for the formulas having no summation: x, f, and (f, x) range over X, F, and $E_{\mathcal{G}}$ respectively. In these two problems, f ranges over F, x ranges over X, and (f, x) ranges over $\mathcal{G}$. Also, we have relaxed the integer LP to a real LP, as all solutions to the integer LP are solutions of the real LP. Note that LP (27) encodes the search for a complete matching of maximum weight for $\mathcal{G}$. By the principle of complementary slackness in linear programming,

$$\begin{array}{l} \text{for respective optima of problems (27) and} \\ \text{(28), } \xi_{fx}{>}0 \text{ if and only if } d_x - c_f = d_{fx}, \end{array} \qquad (29)$$

which is exactly the last constraint of (26). Using this translation into linear programs (27) and (28), it is proved in [17], Thm 3.6, that, if a solution exists to Problem 1, then a unique elementwise smallest solution exists. Based on the above analysis, the following Algorithm 2 (FindOffsets) was proposed in [17] for solving (27,28) and was proved to provide the smallest (real) solution for (28), which happens to be integer.

The reason for using the special iterative algorithm for solving the dual LP (28) is that a standard LP-solving algorithm will return an arbitrary solution, not necessarily the smallest one. The following lemma, whic is repeatedly used in Section 11.1 of [3] about the structural analysis of multimode DAE systems, is an obvious consequence of the linearity of problems (27) and (28):

Lemma 16 *Let $\mathcal{G}$ be a given bipartite graph and let two families of weights $(d^1_{fx})_{(f,x)\in E_{\mathcal{G}}}$ and $(d^2_{fx})_{(f,x)\in E_{\mathcal{G}}}$ be related by $d^2_{fx} = M \times d^1_{fx}$ for every $(f, x) \in E_{\mathcal{G}}$, where M is a fixed positive integer. Then, the offsets of the corresponding Σ-method are also related in the same way: $d^2_x = M \times d^1_x$ for every variable x, and $c^2_f = M \times c^1_f$ for every function f.*

Algorithm 2 FindOffsets ($\mathcal{G}$ is a bipartite graph with weights $\{d_{fx}|(f,x) \in E_{\mathcal{G}}\}$)

1. Solve LP (27), which gives a complete matching $\mathcal{M}$ of maximum weight for $\mathcal{G}$; any method for solving LP can be used.
2. Apply the following iteration until a fixpoint is reached (in finitely many steps), from the initial values $c_f = 0$:
   (a) $\forall x : d_x \leftarrow \max\{d_{fx} + c_f \mid (f,x) \in E_{\mathcal{G}}\}$;
   (b) $\forall f : c_f \leftarrow d_x - d_{fx}$ where $(f,x) \in \mathcal{M}$.

A necessary and sufficient condition for Problem 1 to have a solution is that the set of complete matchings for $\mathcal{G}$ is non-empty. This criterion can be evaluated prior to computing the optimal offsets. This completes our background material on structural analysis.

4.2 The Σ-method for non-square systems

Structural analysis must be extended to non-square systems if we wish to allow for guards being evaluated at the current instant—in this case, enabled/disabled equations are progressively determined along with the progressive evaluation of guards, see Appendix A of [3]. Structural analysis is also a powerful tool for addressing problems other than just the simulation of DAE systems, e.g., the generation of failure indicators for system diagnosis. For both uses, we will need to address other cases than square DAE systems. We develop here an adaptation of the Σ-method to non-square DAE systems. In our development, we could simply copy the original constructions, lemmas, and proofs, of [17], while performing marginal adjustments to handle non-square systems. This is presented next.

If $F = 0$ is a non-square DAE system, then its weighted bipartite graph $\mathcal{G}$ has equation nodes and variable nodes in different numbers, hence the concept of complete matching does not apply. In this case, we consider instead the following weaker notion:

we say that a matching $\mathcal{M}$ is *equation-complete* if it covers all the equation nodes. (30)

No equation-complete matching exists for an overconstrained system $F = 0$, i.e., a system with less variables than equations. This being said, we still consider the primal/dual problems (27) and (28). Again, if $\mathcal{G}$ possesses an equation-complete matching, then problem (27) has a solution, since we can complete this matching to find a feasible solution. Hence, the existence of at least one matching is still used, except that this matching has to be equation-complete instead of complete.

Next, consider respective optima of (27) and (28). They still satisfy the slackness conditions (29), which yields the following results.

- An optimal solution of (27) defines a subgraph $\mathcal{H} \subseteq \mathcal{G}$ by: $(f,x) \in \mathcal{H}$ if and only if $\xi_{fx} = 1$. Note that $\mathcal{H}$ is not a matching, as one equation may be associated to more than one variable in this graph. In contrast, each variable is associated to exactly one equation.
- By the slackness conditions (29), for each equation $f = 0$ and each x such that $(f,x) \in \mathcal{H}$, x occurs in f with differentiation degree equal to the maximum differentiation degree in the entire system, namely d_x.

Call F_Σ the DAE system defined by $\mathcal{H}$ and denote by $Z = \{x^{d_x} \mid x \in X\}$ the set of its leading variables. F_Σ is then regarded as an algebraic system of equations with Z as dependent variables, and we apply the Dulmage-Mendelsohn (DM) decomposition to it. This yields $(\beta_{\mathfrak{U}}, \beta_{\mathfrak{E}}, \emptyset)$, as we know that the overconstrained block $\beta_{\mathfrak{O}}$ is empty.

4.2.1 Justification of the extension

To justify our above adaptation of Pryce's Σ-method, the following questions must be answered:

Question 1 How to adapt Step 2 of Algorithm 2, that is, the fixpoint iteration used for computing the offsets?

Question 2 Is existence and uniqueness of the smallest solution still guaranteed in the non-square case?

Addressing Question 1: Step 2a of Algorithm 2 is not modified. Step 2b, on the other hand, cannot be kept as is, since $\mathcal{H}$ is no longer a matching and there may be more than one x such that $(f,x)\in\mathcal{H}$. Step 2 of Algorithm 2 is thus modified as follows:

(a) $\forall x : d_x \leftarrow \max\{d_{fx} + c_f \mid (f,x) \in E_{\mathcal{G}}\}$;
(b) $\forall f : c_f \leftarrow \max\{d_x - d_{fx} \mid (f,x) \in \mathcal{H}\}$.

We call Algorithm 2′ the resulting algorithm.

Addressing Question 2: This question is answered by the following lemma:

Lemma 17 (non-square Σ-method) Algorithm 2′ *converges to a fixpoint if and only if $\mathcal{H}$ is an optimal solution of the primal problem (27). Let $\mathbf{c}^*, \mathbf{d}^*$ be this fixpoint when it exists. Then $\mathbf{c}^*, \mathbf{d}^*$ is an integer-valued optimal solution for the dual problem (28) and, for any optimal solution $\mathbf{c}^\dagger, \mathbf{d}^\dagger$ for this dual problem, $\mathbf{c}^\dagger \geq \mathbf{c}^*$ and $\mathbf{d}^\dagger \geq \mathbf{d}^*$ hold.*

The last statement expresses the uniqueness of the smallest optimal solution. Lemma 17 was stated and proved in [17], albeit for square DAE systems. Here, we will be carefully following the steps of that proof, while checking that they suit the non-square case.

Proof Following [17], let

$$\phi : \mathbf{c} \mapsto \phi(\mathbf{c})$$

be the mapping corresponding to the successive application of steps (a) and (b) to $\mathbf{c}$.

Assume that $\mathcal{H}$ is an optimal solution of (27) and $\mathbf{c}, \mathbf{d}$ is an optimal solution of its dual problem (28). By slackness conditions (29), one has $d_x = c_f + d_{fx}$ for $(f,x) \in \mathcal{H}$. Starting from $\mathbf{c}$, step (a) yields

$$\begin{aligned} \mathbf{d}(\mathbf{c}) &= \left\{ \max_{(f,x)\in E_{\mathcal{G}}} (d_{fx} + c_f) \,\middle|\, x\in X \right\} \\ &= \{ d_{fx} + c_f \text{ where } (f,x) \in \mathcal{H} \mid x\in X \} \end{aligned}$$

so that, in turn, applying step (b) yields $\mathbf{c}$: hence, $\phi(\mathbf{c}) = \mathbf{c}$ (actually, in step (b), all the $d_x - d_{fx}$ for $(f,x) \in \mathcal{H}$ are equal). In other words, $\mathbf{c}$ is a fixpoint of ϕ.

Conversely, let $\mathcal{H} \subseteq \mathcal{G}$ be any feasible solution of (27), not necessarily optimal, and let $\mathbf{c}^*, \mathbf{d}^*$ be any fixpoint for ϕ (note that ϕ depends on $\mathcal{H}$). We have $d^*_x \geq d_{fx} + c^*_f$ for any $(f,x) \in E_{\mathcal{G}}$, with equality if $(f,x) \in \mathcal{H}$. Assume that $\mathbf{c}^* \geq 0$ also holds; as a matter of fact, $\mathbf{c}^*, \mathbf{d}^*$ is a feasible solution of the dual problem (28). Finally, $\mathcal{H}$ is an optimal solution of the primal problem, and $\mathbf{c}^*, \mathbf{d}^*$ an optimal solution of the dual problem, because they satisfy the slackness conditions (29), which are known to characterize optimal solutions. By contraposition, if $\mathcal{H}$ is not optimal, then no fixpoint of ϕ exists.

It remains to be proved that $\mathbf{c}^* \geq 0$ holds. By construction, ϕ is nondecreasing: $\mathbf{c}' \leq \mathbf{c}$ implies $\phi(\mathbf{c}') \leq \phi(\mathbf{c})$. Denote by $\mathbf{c}^0, \mathbf{d}^0, \mathbf{c}^1, \mathbf{d}^1 \ldots$ the alternating sequence of $\mathbf{c}$'s and $\mathbf{d}$'s obtained by looping over steps (a) and (b), alternatively. We have $\mathbf{c}^k = \phi(\mathbf{c}^{k-1})$ for $k > 0$. If $\mathbf{c}^0 = 0$, then $\mathbf{d}^0 \geq 0$ and $c^1_f = d^0_x - d_{fx} \geq 0$, where x is any variable such that $(f,x) \in \mathcal{H}$ (the value $d^0_x - d_{fx}$ is independent from this choice). As a consequence, $\mathbf{c}_1 \geq 0$. Since ϕ is nondecreasing, applying ϕ to both sides of $\mathbf{c}_1 \geq \mathbf{c}_0 = 0$ yields $\mathbf{c}_2 \geq \mathbf{c}_1$, and so on, so that the algorithm yields a nondecreasing sequence, showing that $\mathbf{c}^* \geq 0$.

Finally, if $\mathbf{c}^\dagger, \mathbf{d}^\dagger$ is another dual-optimal solution for the offsets, then $\mathbf{c}^\dagger$ is a fixpoint of ϕ and thus $\mathbf{c}^\dagger = \phi(\mathbf{c}^\dagger) \geq \phi(0)$, hence $\mathbf{c}^\dagger \geq \phi^k(0)$ for every k, whence $\mathbf{c}^\dagger \geq \mathbf{c}^*$ follows. □

4.2.2 Link with the Pantelides algorithm

In this section, we closely follow Section 5.3 of [17], regarding the comparison of the Σ-method with Pantelides' method [15] for the square case, and adapt it to the non-square case.

Pantelides considers DAE systems with degree at most 1 for all variables. His construction finds a *minimally structurally singular* (MSS) subset of equations and differentiates each equation in this set, creating an augmented system. This is repeated until no more MSS are found. Pantelides writes the system as $n+m$ equations $F(X,Z)=0$, where X stands for the n variables whose derivatives appear, Y stands for the m algebraic variables, and $Z=(X',Y)$. Pantelides' construction applies to this reformulation. For our comparison, the Z coincides with the collection of *leading variables* x^{d_x} where $d_x=\max_f d_{fx}$ is the leading differentiation degree of variable x in the system.

We assume that the primal AP (27) has an optimal solution $\mathcal{H}^*$, and that there exist smallest optimal offsets c_f^*, d_x^* satisfying $\sum_x d_x^* - \sum_f c_f^* = w(\mathcal{H}^*)$ (the total weight of $\mathcal{H}^*$, defined as $\sum_{(f,x)\in\mathcal{H}^*} d_{fx}$). Let

$$\widehat{d}_x =_{\text{def}} \max_f d_{fx} \tag{31}$$

and define the *leading derivative pattern* L of the weighted bipartite graph $\mathcal{G}$ (with weights d_{fx}) as the set

$$L=\{(f,x)\mid d_{fx}=\widehat{d}_x\}$$

For some equation f, respectively some subset E of equations, consider

$$L_f=\{x\mid (f,x)\in L\}\ ,\ \text{and}\ \ L(E)=\bigcup_{f\in E} L_f\,.$$

Following Pantelides, we say that E is *structurally singular (SS)* if $|L(E)|<|E|$. E is *minimally SS (MSS)* if it is SS and has no proper SS subset. The link with the Σ-method is established in the following results.

Lemma 18

1. *If E is MSS with k elements, then $|L(E)|=k-1$*
2. *The following three statements are equivalent:*
   - *(a) L has an equation-complete matching, see (30);*
   - *(b) There are no SS subsets;*
   - *(c) All c_i are equal to 0.*

Proof For statement 1, if $L(E)$ has fewer than $k-1$ elements, then removing any element from E yields a set E' satisfying $|L(E')|\le|L(E)|<k-1=|E'|$. Therefore, E' is SS, which contradicts the minimality of E.

The equivalence 2a⇔2b is just Hall's Theorem applied to the sets L_e.

2a⇒2c: If L has an equation-complete matching $\mathcal{H}$, then (by definition of L) for $(f,x)\in\mathcal{H}$, $d_{fx}=\widehat{d}_x$. In other words, d_{fx} is maximal among all the $d_{\widehat{f}x}$ for $\widehat{f}$ ranging over the set of all equations. This means that $\mathcal{H}$ is optimal for the primal problem. We have $d_x^*-c_f^*=d_{fx}=\widehat{d}_x$ for $(f,x)\in\mathcal{H}$ hence $c_f^*=0, d_x^*=\widehat{d}_x$ is the (unique) smallest optimal solution for the dual problem.

¬2a⇒ ¬2c: If 2a fails to hold, then the optimal solution $\mathcal{H}^*$ for the primal problem contains a (f,x) with $d_{fx}<\widehat{d}_x$. Then, by the slackness condition (29):

$$d_x^*=d_{fx}+c_f^*<\widehat{d}_x+c_f^*\,. \tag{32}$$

On the other hand, since d^*_y, c^*_f is a dual-optimal solution, we have $d^*_y \geq d_{fy} + c^*_f \geq d_{fy}$ for all y, whence

$$d^*_y \geq \widehat{d}_y \,. \tag{33}$$

In particular, $d^*_y \geq \widehat{d}_y$; combining this result with (32) yields $c_f > d^*_x - \widehat{d}_x \geq 0$, so that 2c is false. □

Lemma 19 *Let $\mathcal{H}^*$ be a primal-optimal solution as before. If $c^*_{f_o} = 0$ is satisfied for some equation f_o, then*

1. *Let x_o be such that $(f_o, x_o) \in \mathcal{H}^*$, then $(f_o, x_o) \in L$;*
2. *$c_f = 0$ holds for any $f \neq f_o$ such that $(f, x_o) \in L$.*

Proof We have

$$\begin{array}{rcl} d_{f_o x_o} & = & d^*_{x_o} \text{ (since } d^* \text{ is optimal and } c^*_{f_o} = 0) \\ & \geq & \widehat{d}_{x_o} \text{ (by (33))} \\ & \geq & d_{f_o x_o} \text{ (by (31))} \end{array}$$

hence we get

$$d^*_{x_o} = \widehat{d}_{x_o} \tag{34}$$

and $d_{f_o x_o} = \widehat{d}_{x_o}$, which proves statement 1. For statement 2, if $(f, x_o) \in L$ for some f, then

$$0 = \widehat{d}_{x_o} - d_{f x_o} = d^*_{x_o} - d_{f x_o} \geq c^*_f \geq 0$$

whence $c^*_f = 0$. □

Lemma 20 *For E any MSS, we have $\forall f \in E : c_f > 0$.*

Proof Suppose the lemma is false, meaning that $E_0 =_{\text{def}} \{f \in E \mid c_f = 0\}$ is nonempty. Let $|E| = k$ and $|E_0| = l$, so that $1 \leq l \leq k$. By statement 1 of Lemma 19, we have $L(E_0) \geq l$. Since $|L(E)| < |E|$ by the assumption that E is an MSS, we deduce $l < k$, hence $E \setminus E_0$ is nonempty. Pick any $f \in E \setminus E_0$, then $c_f > 0$. By statement 2 of Lemma 19, L_f cannot contain any x that is reachable from E_0 via $\mathcal{H}^*$:

$$L(E \setminus E_0) \cap [\exists f. ((E_0 \times X) \cap \mathcal{H}^*)] = \emptyset \tag{35}$$

The set on the right-hand side of $\cap$ is contained in $L(E)$ (by statement 1 of Lemma 19) and has cardinality at least equal to $|E_0|$. By (35), we get $|L(E \setminus E_0)| \leq |L(E)| - |E_0| < |E \setminus E_0|$, hence $E \setminus E_0$ is a SS subset strictly contained in E, thus contradicting the minimality of E. □

Following these results, Pantelides' algorithm can be extended to non-square systems in pretty much the same way the Σ-method was extended. This non-square Pantelides' algorithm locates MSS subsets and differentiates them, until there is no such subset left. Note that, when Pantelides' algorithm locates a MSS subset I and differentiates it, this amounts to modifying the offsets found by the Σ-method in the following way:

- the d_j remain unchanged
- the c_i decrease by 1 if $i \in I$, remain unchanged otherwise.

From this and the preceding results, one gets:

Theorem 21 (equivalence with Pantelides) *Pantelides' algorithm gives the same results as the Σ-method, in that the same offsets c_i are found.*

4.3 Differential and difference arrays

Differential arrays were proposed in [7] as a tool for finding the latent equations of a DAE system. To $F = 0$ a DAE system, we associate its k-th *differential array* $\mathcal{A}_k$ defined, for any $k \in \mathbb{N}_{\geq 0}$, by

$$\mathcal{A}_k \quad =_{\text{def}} \quad \begin{bmatrix} F \\ \frac{d}{dt}F \\ \frac{d^2}{dt^2}F \\ \vdots \\ \frac{d^k}{dt^k}F \end{bmatrix} \tag{36}$$

where $\frac{d}{dt}$ denotes the total time derivative. Thus, for k sufficiently large, array system $\mathcal{A}_k = 0$ should contain all the latent equations that must be added to $F = 0$ in order to reduce its index to 0. More precisely, the index reduced counterpart of $F = 0$ is the following system with existential quantifiers:

$$\exists W : \mathcal{A}_k(X, W, Y) = 0 \tag{37}$$

where:

- Y collects the free variables of system $F = 0$;
- X collects the dependent variables of system $F = 0$;
- W collects other supplementary variables involved in array $\mathcal{A}_k$.

Difference arrays are defined similarly for discrete-time dAE systems, by replacing the j-th time derivatives $\frac{d^j}{dt^j}F$ by the j-th forward shift $F^{\bullet j}$ defined in (24), when building array (36):

$$\mathcal{A}_k \quad =_{\text{def}} \quad \begin{bmatrix} F \\ F^{\bullet} \\ F^{\bullet 2} \\ \vdots \\ F^{\bullet k} \end{bmatrix} \tag{38}$$

The structural analysis of systems with existential quantifiers, of the form (19), was studied in Section 3.4. We can apply it to the structural analysis of arrays (36) or (38) to find the latent equations of a DAE system, and, thus, its index.

The resulting algorithm is much less efficient than Pryce's Σ-method or Pantelides algorithm. The reason is that, unlike the above methods, the one of Section 3.4 does not exploit the fact that the rows of the array are successive shifts of the same dAE system. In turn, *the method of Section 3.4 extends to time-varying discrete-time systems.* We just need to replace, in (38), the shifted versions of F by the successive $F(t_0), \ldots, F(t_k)$ at successive discrete instants $t_0, \ldots, t_k$. This is used in [2, 3] to develop a structural analysis of finite cascades of mode changes.

5 Summary of results of [2, 3] regarding multimode DAE systems

In this section, we briefly review the results of [2, 3] regarding multimode DAE systems simulation. We indicate where, in [2, 3], the developments of Sections 3 and 4 of this report are used.

5.1 Multimode DAE and dAE systems definition

In [2, 3], a mathematical definition for multimode DAE systems is provided, and we also define their discrete-time counterpart, namely multimode dAE systems, of the respective forms

$$\begin{array}{rrcl} \text{multimode DAE:} & \text{if } \gamma_j(x_i\text{'s and derivatives}) & \text{then} & f_j(x_i\text{'s and derivatives}) = 0 \\ \text{multimode dAE:} & \text{if } \gamma_j(x_i\text{'s and shifts}) & \text{then} & f_j(x_i\text{'s and shifts}) = 0\,, \end{array} \tag{39}$$

where $x^{\bullet k}$, the k-shift of x, is defined in (24).

5.2 Nonstandard semantics

To allow for a uniform handling of modes and mode change events in the structural analysis of multimode DAE, we use a *nonstandard semantics* for DAEs and multimode DAEs. In this nonstandard semantics,

$$\text{derivative } x' \text{ is interpreted as its first-order forward Euler scheme } \tfrac{x^\bullet - x}{\partial}, \tag{40}$$

where continuous-time is discretized by using an *infinitesimal* positive time step ∂ and the forward shift $x^\bullet$ is defined in this discrete-time. Here, as already stated at the beginning of Section 2.3,

> "infinitesimal" means "smaller than any positive real number", which can be given a formal meaning in *nonstandard analysis* [9, 18].

This mapping allows for a discretization of multimode DAE systems with an infinitesimal error, which yields a faithful discrete-time approximation. Of course, there is no free lunch: the nonstandard semantics is not effective in that there is no computer that can execute it. Still, it is highly useful for the structural analysis, which is a symbolic analysis. The nonstandard semantics of a multimode DAE systems is a multimode dAE system, in which

- occurrences of increments $\frac{x^\bullet - x}{\partial}$ captures the continuous-time dynamics of the considered DAE (for this reason, we call *continuous* the modes in which such dynamics occurs), whereas
- *mode changes* are represented by transitions relating the value x^- of a state just before the change and its value x^+ right after the change, encoded in the nonstandard semantics as x and $x^\bullet$.

5.3 Structural analysis of multimode DAE/dAE systems

The structural analyses of both DAE and dAE systems coincide, by replacing the differentiation operator $x \mapsto x'$ arising in DAEs, by the forward shift operator $x \mapsto x^\bullet$ arising in dAEs. Pryce's Σ-method applies to both DAE and dAE; for the latter, the equation offsets c_f indicate how many times each equation f must be shifted.

5.3.1 Handling continuous modes

Since structural analyses mirror each other in continuous and discrete-times through the mapping (40), the structural analysis of continuous modes is performed as usual, with the help of the Σ-method presented in Section 4. This is summarized in the following

Basic tool 3 *Standard structural analysis can be performed for each continuous mode, in order to add the needed latent equations. The continuous-time structural analysis (where latent equations are found by differentiation) is therefore used to generate code within continuous modes.*

5.3.2 Handling mode changes

Basic tool 4 *In order to prepare for the generation of restart code at mode changes, the discrete-time structural analysis (where latent equations are found by shifting) based on the nonstandard semantics is applied to the continuous modes before and after the change.*

Basic tools 3 and 4 rely on J. Pryce's Σ-method. Once latent equations have been added for each continuous mode, we may have a conflict, at mode change events separating two successive continuous modes, between:

- the next values of states predicted by the dynamics in the previous mode; and
- the consistency equations, generated by the structural analysis of the new mode.

Is this pathological? Not quite. Such a situation can occur even for physically meaningful models. If the new continuous mode has positive index, then, nontrivial consistency equations are generated by the structural analysis, which may be conflicting with the dynamics of the continuous mode before the change. Since one cannot regard this model as being incorrect, we need a policy for handling these conflicts.

Handling isolated mode changes separating two successive continuous modes: In [2, 3], it is proposed to call causality to the rescue: for isolated mode changes, we give priority to the previous dynamics and postpone for a while the conflicting consistency equations generated by the new continuous mode. Here, "for a while" means "for a finite number of infinitesimal time steps", which amounts to zero time in real duration.

Basic tool 5 *Identifying the conflicting consistency equations at mode changes is performed by using the Dulmage-Mendelsohn decomposition (Definition 7 and associated Lemmas 8 and 9).*

Conflicting consistency equations of the new continuous mode are postponed for a finite number of nonstandard instants. The resulting discrete-time system operates for finitely many nonstandard instants, and is the basis for recovering the restart values of the states in the new mode. This establishes the structural analysis of mode changes separating two successive continuous modes.

Handling finite cascades of mode changes: We also handle in [2, 3] finite cascades of successive mode changes separating two successive continuous modes. Since the dynamics varies over the successive instants of the cascade, time-invariance no longer holds, and, thus, the Σ-method does not apply. However, the method of *difference arrays,* closely derived from Campbell and Gear's *differential arrays* [7], can be adapted to time-varying discrete-time systems, and invoked. See Section 4.3 and formula (38) for a short introduction to difference arrays.

For time-invariant dynamics, difference arrays are obtained by stacking, one below the other, the dAE dynamics and its successive shifted versions. For a finite cascade of mode changes leading to a new continuous mode, however, we form the associated difference array by stacking, one below the other, the dynamics that holds at each successive instant of the cascade. More precisely, let t be the current instant in the nonstandard semantics, and let ∂ be the infinitesimal time step. Then, the time-varying array associated to the cascade of mode changes is

$$\mathcal{A}_k(t) \quad =_{\text{def}} \quad \begin{bmatrix} F(t) \\ F(t+\partial) \\ F(t+2\partial) \\ \vdots \\ F(t+k\partial) \end{bmatrix} \tag{41}$$

where $F(t)$ is the dynamics at the first event of the cascade, $F(t+\partial)$ is the dynamics at the second event of the cascade, and so on until the instant $t+k\partial$, which sits within the new continuous mode. Enough rows should be added to the array, so that the leading variables of the first event of the

cascade are uniquely determined as functions of the states before the change. As for differential arrays, the extra variables brought by the successive shifting must be eliminated by existential quantification—see the comments regarding formula (11) introducing differential arrays in [7]. Finite cascades of mode changes are thus handled by using the results of Section 3.4 regarding equations with existential quantifiers:

Basic tool 6 *Finite cascades of mode changes are handled by forming the* difference array (41) *and analyzing it by using the method of Section 3.4 regarding equations with existential quantifiers.*

Fixpoint between guards and the equations they control: In a multimode DAE system, a logico-numerical fixpoint occurs when a variable x is determined by a different dynamics depending on the value of a guard that itself depends, directly or indirectly, on x. In [2, 3], only multimode DAE systems having no logico-numerical fixpoint are supported. In such models, the above guard should rather depend on the left-limit x^- of x, where $x^-(t) = \lim_{s \nearrow t} x(s)$. Nevertheless, in Appendix A of [3], we propose a method for handling multimode DAE systems with fixpoints but no cascades of length > 1. This method uses the following basic tool:

Basic tool 7 *The Σ-method for non-square systems, presented in Section 4.2.*

5.3.3 Generic form of the generated simulation code

For S a multimode DAE model having continuous modes separated by finite cascades of mode change events, the method proposed in [3] returns the following:

Case of a structurally incorrect model: The compilation algorithm returns, for each mode and mode change event in which structural analysis fails, the overdetermined subsets of equations and the underdetermined subsets of variables.

Case of a structurally correct model, producing an *mDAE interpreter*: The compilation algorithm returns an *mDAE interpreter,* which is a labeled bipartite graph $\mathcal{G}$ whose vertices are:

- state and leading variables x of the system, as well as guards γ;
- structurally nonsingular blocks β of algebraic equations determining leading variables from state variables, and expressions *exp* determining the values of guards as function of state and leading variables.

A branch $x \to \beta$ exists in $\mathcal{G}$ if x is a free variable (input) of block β and a branch $\beta \to x$ exists in $\mathcal{G}$ if x is a dependent variable (output) of block β. A branch $x \to exp$ exists in $\mathcal{G}$ if *exp* has x as one of its arguments, and a branch $exp \to \gamma$ exists in $\mathcal{G}$ if γ is computed using expression *exp*.

Not every block is involved in a given mode or mode change. Thus, to each block β we associate a *label* $\lambda(\beta)$, which is a property characterizing the set of all modes or mode changes in which β is active. A label is then assigned to every *path* π traversing blocks $\beta_1, \ldots, \beta_k, \ldots, \beta_K$ by setting $\lambda(\pi) = \bigwedge_1^K \lambda(\beta_k)$. Every circuit of $\mathcal{G}$ has label F, which ensures that the code is not globally circular. Also, for any two blocks $\beta_1 \neq \beta_2$ leading to the same variable x in $\mathcal{G}$, we have $\lambda(\beta_1) \wedge \lambda(\beta_2) = \text{F}$, which ensures that two different blocks never compete at determining the same variable x. This interpreter encodes all the different schedulings that are valid for the different modes of the system. Note that modes are not enumerated, which is essential in order for this technique to scale up.

This technique is reminiscent of the *conditional dependency graphs* used in the compilation of the Signal synchronous language [1, 4].

The notion of mDAE interpreter is illustrated in Figure 5, for an RLDC2 circuit example with schematic shown in Figure 3 and model given in System (42), see Section 12 of [3] for details. This form of mdAE interpreter is used in the IsamDAE tool [6], by representing all the label using BDD-related data structures.[7]

[7] BDD means: Binary Decision Diagram, a data structure developed for model checking.

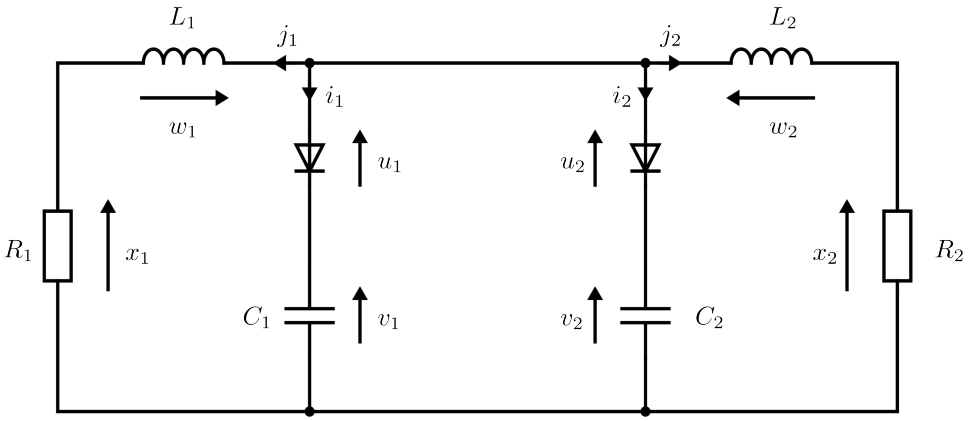


Figure 3: Schematics of the RLDC2 circuit.

$$
\begin{array}{rcll}
0 & = & i_1 + i_2 + j_1 + j_2 & (K_1) \\
x_1 + w_1 & = & u_1 + v_1 & (K_2) \\
u_1 + v_1 & = & u_2 + v_2 & (K_3) \\
u_2 + v_2 & = & x_2 + w_2 & (K_4) \\
w_1 & = & L_1 \cdot j_1' & (L_1) \\
w_2 & = & L_2 \cdot j_2' & (L_2) \\
i_1 & = & C_1 \cdot v_1' & (C_1) \\
i_2 & = & C_2 \cdot v_2' & (C_2) \\
x_1 & = & R_1 \cdot j_1 & (R_1) \\
x_2 & = & R_2 \cdot j_2 & (R_2) \\
s_1 & = & \text{if } \gamma_1 \text{ then } i_1 \text{ else } -u_1 & (S_1) \\
s_2 & = & \text{if } \gamma_2 \text{ then } i_2 \text{ else } -u_2 & (S_2) \\
0 & = & \text{if } \gamma_1 \text{ then } u_1 \text{ else } i_1 & (Z_1) \\
0 & = & \text{if } \gamma_2 \text{ then } u_2 \text{ else } i_2 & (Z_2) \\
\gamma_1 & = & (s_1^- \geq 0) & (\gamma_1^-) \\
\gamma_2 & = & (s_2^- \geq 0) & (\gamma_2^-)
\end{array}
\tag{42}
$$

Figure 4: The RLDC2 model (*n.b.*: variable y' denotes the time derivative of y).

6 Conclusion

This report complements references [2, 3] by collecting and providing all missing details related to structural analysis, for both algebraic and DAE systems, in a systematic way.

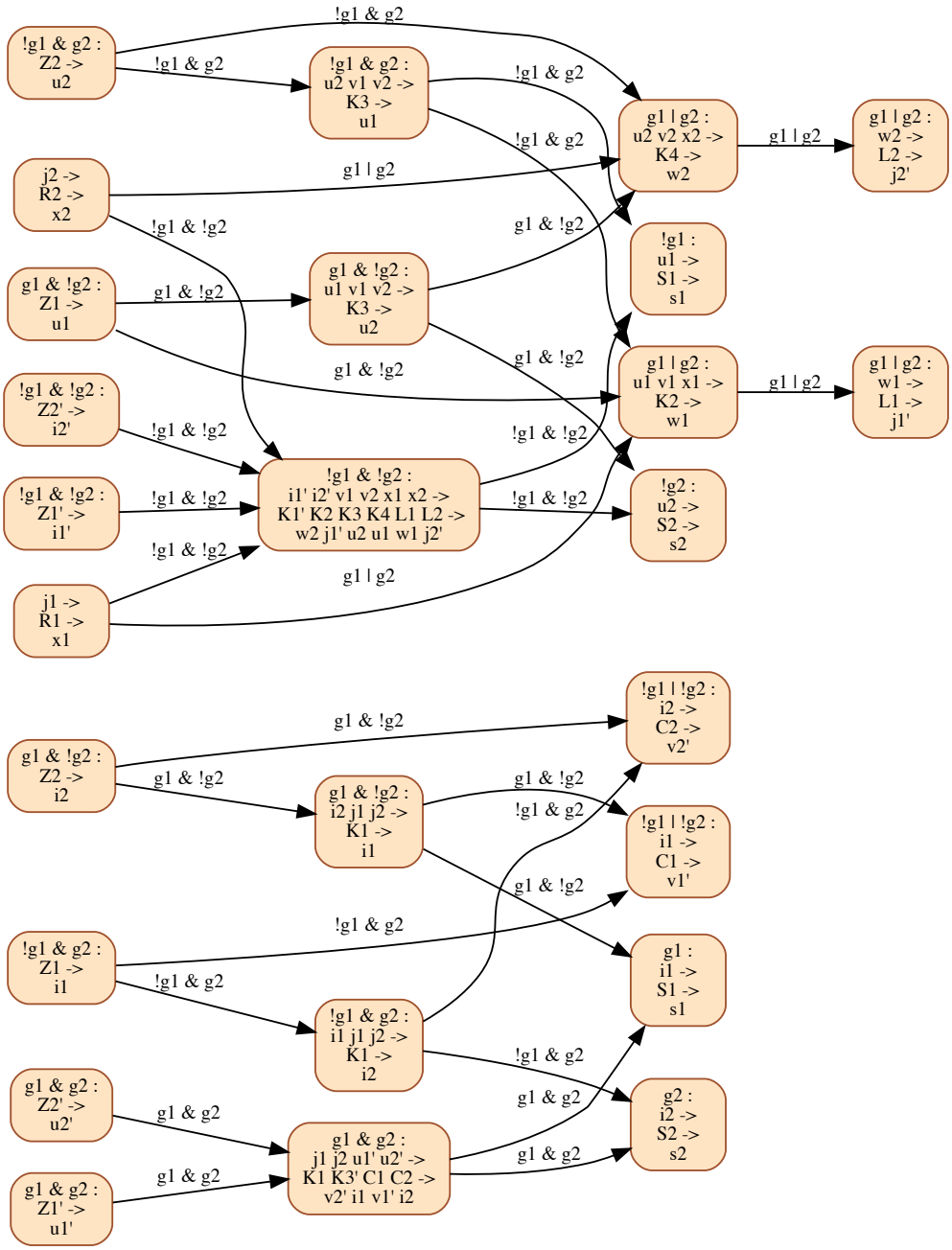


Figure 5: Block dependency graph of the RLDC2 model, generated by IsamDAE. Vertices are labeled $p : Y \to \beta \to X$, where: p is a propositional formula defining in which modes the block is evaluated; Y is the set of free variables for reading; β is the set of equations of the block; X is the set of dependent variables for writing. Edges are labeled by a propositional formula, defining in which modes the dependency applies—notation "!g" means "not g", "g1 & g2" is the conjunction, and "g1 | g2" is the disjunction.

References

[1] A. Benveniste, B. Caillaud, and P. L. Guernic. Compositionality in dataflow synchronous languages: specification & distributed code generation. *Information and Computatin*, 163:125–171, 2000.

[2] A. Benveniste, B. Caillaud, and M. Malandain. The mathematical foundations of physical systems modeling languages. *Annual Reviews in Control*, 2020.

[3] A. Benveniste, B. Caillaud, and M. Malandain. The mathematical foundations of physical systems modeling languages. *CoRR*, abs/2008.05166, 2020.

[4] A. Benveniste, P. Caspi, S. Edwards, N. Halbwachs, P. Le Guernic, and R. de Simone. The synchronous languages 12 years later. *Proceedings of the IEEE*, 91(1), Jan. 2003.

[5] C. Berge. *The theory of graphs and its applications*. Wiley, 1962.

[6] B. Caillaud, M. Malandain, and J. Thibault. Implicit structural analysis of multimode DAE systems. In *23rd ACM International Conference on Hybrid Systems: Computation and Control (HSCC 2020)*, Sydney, Australia, April 2020. to appear.

[7] S. L. Campbell and C. W. Gear. The index of general nonlinear DAEs. *Numer. Math.*, 72:173–196, 1995.

[8] H. Cartan. *Formes Différentielles*. Collection Méthodes. Hermann, 1967.

[9] N. Cutland. *Nonstandard analysis and its applications*. Cambridge Univ. Press, 1988.

[10] J. Dieudonné. *Fondements de l'analyse moderne*. Gauthier-Villars, 1965.

[11] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Numerical Mathematics and Scientific Computation. Oxford University Press, 1986.

[12] L. Hogben, editor. *Handbook of Linear Algebra*. CRC Press, Boca Raton, FL, USA, 2006.

[13] L. Hogben. *Handbook of Linear Algebra*. 2nd edition, 2014.

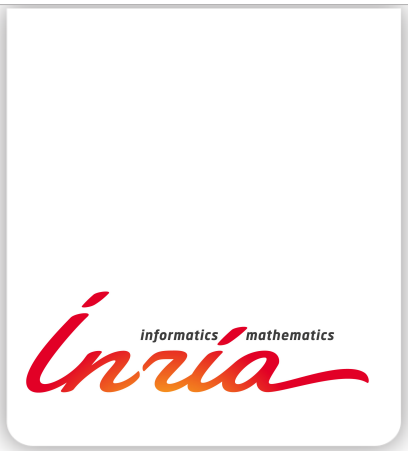
[14] S. E. Mattsson and G. Söderlind. Index reduction in Differential-Algebraic Equations using dummy derivatives. *Siam J. Sci. Comput.*, 14(3):677–692, 1993.

[15] C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM J. Sci. Stat. Comput.*, 9(2):213–231, 1988.

[16] A. Pothen and C. Fan. Computing the block triangular form of a sparse matrix. *ACM Trans. Math. Softw.*, 16(4):303–324, 1990.

[17] J. D. Pryce. A simple structural analysis method for DAEs. *BIT*, 41(2):364–394, 2001.

[18] A. Robinson. *Nonstandard Analysis*. Princeton Landmarks in Mathematics, 1996. ISBN 0-691-04490-2.



RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-0803